Koordination und Kooperation von Mehrrobotersystemen unter spatialen Nebenbedingungen

Dissertation zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

Vorgelegt von
Bernd Brüggemann
aus Wickede (Ruhr), Deutschland

Bonn, 2014

Dissertation

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universiät Bonn

Erstgutachter (Betreuer): Prof. Dr. Peter Martini
Zweitgutachter: PD Dr. Elmar Langetepe

Tag der mündlichen Prüfung: 11.08.2014
Erscheinungsjahr: 2014

Zusammenfassung

Schon die Steuerung eines einzelnen Roboters ist komplex und erfordert oft ein intensives Training. Soll ein System aus mehreren Robotern kontrolliert werden, potenziert sich diese Schwierigkeit. Zudem müssen neben dem eigentlichen Ziel, das das Mehrrobotersystem (MRS) verfolgt, häufig zusätzliche Nebenbedingungen eingehalten werden, die mit der eigentlichen Aufgabe nur am Rande zu tun haben. Daher wird in dieser Arbeit ein neues Planungsverfahren vorgestellt, welches, nach Vorgaben des Benutzers und unter Berücksichtigung einer spatialen Nebenbedingung, geeignete Endkonfigurationen und den Weg dorthin vorschlägt.

Diese Arbeit stellt sowohl die theoretischen Grundlagen als auch die praktischen Umsetzungen der koordinierten Navigation unter Nebenbedingung vor. Der Schwerpunkt liegt in der Erstellung sogenannter globaler Mehrroboterpläne. Diese enthalten Zielpunkte für jeden Roboter sowie die Wege zwischen den Zielpunkten. Zur Berechnung der globalen Mehrroboterpläne wird die Umwelt diskretisiert und die Bewegungsmöglichkeiten sowie die Nebenbedingung in je einem Graphen dargestellt. Diese beiden Graphen zeigen, wo die Roboter hinfahren können und wo sie hinfahren dürfen. Durch die graphenbasierte Darstellung der Umgebung und der Nebenbedingung kann dabei weitestgehend von der konkreten Aufgabe des Mehrrobotersystems abstrahiert und somit eine große Gruppe von Nebenbedingungen betrachtet werden.

Mit der Darstellung des globalen Planungsproblems unter spatialen Nebenbedingungen als Graphenproblem werden zwei Algorithmen vorgestellt, die die globale Planung durchführen können. Der Algorithmus STPlaner nutzt die Äquivalenz zwischen der Endkonfiguration und dem Steinerbaum-Problem aus, während der AgentPlaner an den MPR-Flooding Algorithmus angelehnt ist. Zudem wird eine Variante des AgentPlaners, der FastAgentPlaner, vorgestellt.

Die Eigenschaften und die Performance des STPlaner sowie des AgentPlaner und des FastAgentPlaner werden in ausführlichen Simulationen getestet. Dabei dient der STPlaner als Referenzalgorithmus, da er, aufgrund der großen Nähe zu den bekannten Steinerbaum-Heuristiken, die Bewertung des AgentPlaner und des FastAgentPlaner vereinfacht. Es zeigt sich bei der Auswertung, dass, obwohl die Endkonfigurationen des STPlaner weniger Roboter benötigen als die anderen Algorithmen, die Pläne des AgentPlaners für eine reale Umsetzung besser geeignet sind. Nach der Betrachtung des Problems in einer statischen Umgebung wird das Verhalten des AgentPlaners in dynamischen Umgebungen untersucht. Dabei werden zwei Fälle untersucht: im Voraus bekannte Graphen, bei denen während der Planausführung Kanten gelöscht werden, sowie Graphen, die zunächst unbekannt sind und erst erkundet werden müssen. Durch die Übertragung des Planungsverfahrens auf reale Systeme wird zuletzt in zwei beispielhaften Versuchen mit mehreren Robotern gezeigt, dass das Planungsverfahren auch in realen Umgebungen eingesetzt werden kann.

At first glance motion planning looks relatively simple, since humans deal with it with no apparent difficulty in their every day lives.

Jean-Claude Latombe

Danksagung

Auch wenn die Dissertation „die selbstständige wissenschaftliche Arbeit“ (Nach „Promotionsordnung der Mathematischen-naturwissenschaftlichen Fakultät“) darstellt, so gibt es viele Personen, die mich auf dem Weg zur Promotion mit Rat und Tat begleitet haben. Ich möchte diese Stelle dazu nutzen, mich bei einigen namentlich, und bei vielen allgemein zu bedanken. Zunächst also „Danke“ an all diejenigen, die mir mit Anmerkungen, Kommentaren und Zuspruch geholfen und mich motiviert haben.

Bedanken möchte ich mich bei Herrn Professor Martini, der diese Arbeit als betreuender Hochschullehrer angenommen hat, sowie den Mitgliedern der Prüfungskommission PD Langetepe, Professor Weber und Professor Schade, dass sie ihre Zeit opferten um meiner Promotion zum Erfolg zu verhelfen. Ganz besonders möchte ich mich bei Dr. Dirk Schulz bedanken, der mir die Freiräume ermöglichte, die in dieser Arbeit beschriebenen Zusammenhänge zu erforschen, jedoch mit kritischen Nachfragen an den richtigen Stellen dafür sorgte, dass die Arbeit fokussiert blieb. Nach meiner Erfahrung während der Erstellung der Dissertation können Gedanken und Ideen nur in der Diskussion mit Kollegen reifen. Hier möchte ich vor allem Michael Brunner und Dennis Wildermuth für die vielen Stunden Geduld und Gespräche danken.

Neben der fachlichen Begleitung ist der Rückhalt in der Familie unerlässlich, um so ein Jahre währendes und zeitintensives Projekt erfolgreich zum Abschluss zu bringen. Ich danke meiner Frau Claudia Brüggemann für die Unterstützung, Motivation und Geduld die nötig waren, sowie Ben und Maja, dass sie bereit waren den Papa auch mal am Schreibtisch zu lassen. Zudem danke ich meiner Mutter Angelika Brüggemann, die leider die Fertigstellung dieser Arbeit nicht mehr erleben konnte.

Abschließend möchte ich meine Hoffnung äußern, dass die Promotion mir als ein Werkzeug dienen kann, weiterhin spannende, abwechslungsreiche und interessante Arbeit tun zu können und möchte mich dafür bedanken, diese Chance bekommen zu haben.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Innerhalb des letzten Jahrzehnts zeigten sich immer neue mögliche Anwendungsgebiete für Roboter. Fortschritte im Bereich der künstlichen Intelligenz scheinen den Einsatz von Haushaltsrobotern langsam in greifbare Nähe zu rücken. Solche Systeme sind dafür gedacht, innerhalb einer schnell alternden Gesellschaft dem Menschen Aufgaben abzunehmen, die er nicht mehr selber erledigen kann oder will. Dafür werden Konstruktionsprinzipien entwickelt, die zu einem ausgeklügelten, hochspezialisierten Roboter führen. So soll ein einziges System mit Fähigkeiten ausgestattet werden, die es ihm ermöglichen, menschliche Aufgaben in einer für Menschen gemachten Umgebung zu erfüllen.
Neben den Aufgabenbereichen, die dem Erhalt des Lebensstandards zugeordnet werden können, wächst auch das Bedürfnis, Roboter für Aufgaben einzusetzen, die im Rahmen von Extremsituationen zu erfüllen sind. Dies sind Aufgaben in Reaktion auf Katastrophen oder Konfliktfällen. In einer Zeit, in der Katastrophen, wenn vielleicht nicht in ihrer Anzahl, so doch in ihrer Heftigkeit zunehmen, sind Roboter eine Möglichkeit, Leben zu retten und Gegenmaßnahmen zu ergreifen, ohne dabei zusätzliche Menschenleben zu gefährden.
Im Gegensatz zu den zuvor erwähnten Servicerobotern setzt sich in diesem Anwendungsfeld die Auffassung durch, dass der Einsatz eines einzelnen, omnipotenten Systems nicht sinnvoll ist. Gerade weil die Aufgaben gefährlich sind, ist Redundanz notwendig und die Erfüllung der Aufgabe sollte nicht von einem einzelnen Roboter abhängen. Daher ist in den letzten Jahren die Programmierung von Mehrrobotersystemen (MRS) immer mehr in den Fokus der Forschung gerückt. Dabei bieten Mehrrobotersysteme ge-

rade in der Gefahrenabwehr entscheidende Vorteile. So ist neben der Redundanz, also der Fähigkeit auch Ausfälle bis zu einem gewissen Grad kompensieren zu können, die Möglichkeit vorhanden, verschiedene spezialisierte Roboter einzusetzen. Dadurch muss nicht jeder Roboter mit jedem Sensor ausgerüstet sein und nur einige der Roboter müssen bestimmte Fähigkeit besitzen, wie z.B. Treppen steigen zu können. Die Komplexität und damit auch die Fehleranfälligkeit des einzelnen Systems nehmen dadurch ab. Zusätzlich kann der Einsatz eines MRS je nach Aufgabe einen Zeitgewinn bedeuten. Indem die Roboter parallel verschiedene Teilaufgaben erfüllen, ist die Gesamtausführungszeit niedriger.

Da kein Vorteil ohne erhöhten Aufwand gewonnen werden kann („no free lunch“, Wolpert and Macready [1997]), bringt der Einsatz eines Mehrrobotersystems allerdings auch Probleme mit sich. Um die Vorteile eines MRS ausnutzen zu können, sollten die einzelnen Roboter geschickt eingesetzt werden. Die Roboter müssen kooperieren und koordiniert werden. Wo allerdings schon Planungsprobleme mit einem einzelnen Roboter eine hohe Komplexität erreichen, bedeuten mehrere Roboter häufig eine exponentielle Vergrößerung der Möglichkeiten und damit des Suchraums. Hier greifen Planungsalgorithmen für Mehrrobotersysteme.

Unterschiedliche Mechanismen zur Planung von Bewegungen und Aktionen in Mehrrobotersystemen sind schon untersucht worden. So sind viele Arbeiten über die Verteilung von Aufgaben (Task Allocation) bekannt, ebenso wie Kooperationsverfahren, die bei der Zuteilung von beschränkten Ressourcen (wie z.B. dem verfügbaren Platz) unterstützen. Auch der Bewegungsaspekt des gesamten MRS, z.B. in einer Formationsfahrt, ist Gegenstand der Forschung. Einen großen Teil der Aufgaben, die einem MRS gestellt werden, macht dabei die Navigation der einzelnen Roboter aus. Eine geplante Navigation für das gesamte MRS, bei der sich die einzelnen Roboter gegenseitig unterstützen, ist allerdings ein wenig untersuchtes Gebiet.

Es zeigt sich, dass oft neben dem eigentlichen Ziel, d.h. der Intention, mit der das MRS gesteuert wird, Nebenbedingungen eingehalten werden müssen, die mit der eigentlichen Aufgabenerfüllung nur am Rande zu tun haben. So wird z.B. bei Aufgaben zur Informationsgewinnung in Katastrophengebieten implizit angenommen, dass die einzelnen Roboter Kontakt mit der Stelle haben, an der die Informationen benötigt werden. In einer großen oder stark bebauten Umgebung ist solch eine Annahme aber nicht haltbar. Sind die zu gewinnenden Informationen aber zeitkritisch, ist eine ständige Funkverbindung unerlässlich und damit eine notwendige Nebenbedingung bei der Erkundung des Geländes.

Diese Arbeit stellt sowohl die theoretischen Grundlagen als auch praktischen Umsetzungen der koordinierten Navigation unter Nebenbedingung vor. Die Ergebnisse ermöglichen die Erstellung eines Frameworks, welches für ein MRS einen Navigationsplan entwirft, bei dem Nebenbedingungen eingehalten werden. Dabei wird gezeigt, dass es möglich ist, das Framework von der Ebene der einzelnen Aufgabenerfüllung so weit zu abstrahieren, dass ein Planungsverfahren entsteht, welches unabhängig ist von:

- der Roboterbewegung,
- der Umgebung,
- der Aufgabe sowie
- der Art der Nebenbedingung.

1.2 Problembeschreibung

Schon die Kontrolle eines einzelnen Roboters ist für einen Operateur schwierig und erfordert meist intensives Training. Soll nun ein Operateur eine Gruppe von Robotern gezielt steuern, ist dieser schnell an der Belastungsgrenze. Daher wurde schon in [Arai et al., 2002] die Frage gestellt, wie man einem Benutzer die Kontrolle eines Mehrrobotersystems ermöglichen kann.

1.2.1 Koordinierte Navigation unter spatialer Nebenbedingung

Gerade in der Gefahrenabwehr und im Katastrophenmanagement kann es sinnvoll sein, Roboter an verschiedenen Stellen in der Umgebung zu platzieren. Diese Punkte werden vom Benutzer ausgesucht. Daraufhin müssen die Roboter einen Weg finden, um diese Zielpunkte einzunehmen. Um jedoch als Benutzer im Leitstand die Daten der Roboter zu empfangen, muss das Mehrrobotersystem zudem die Kommunikation zwischen den Robotern aufrecht erhalten. Dazu können weitere Roboter in der Umgebung platziert werden, die als Relaisstationen dienen. Da der Benutzer, gerade in Katastrophenszenarien, immer die Kontrolle über das MRS behalten soll und im Vorhinein die geplanten Aktionen kennen muss, wird hier auf eine Offline-Planung gesetzt, die vorher die notwendigen Pfade und Positionen ermittelt.
In dem oben skizzierten Szenario ist die Kommunikation eine sogenannte Nebenbedingung. Das eigentliche Ziel, das für das MRS vorgegeben wurde, ist die Datensammlung an den vom Benutzer definierten Punkten. Um diese Daten jedoch zeitnah übermitteln und damit nutzen zu können, muss die Kommunikation aufrecht erhalten werden. So bildet die Kommunikation eine weitere Bedingung an die Steuerung des Mehrrobotersystems. In dieser Arbeit werden Nebenbedingungen allgemeiner gefasst. Neben der Kommunikation werden alle Nebenbedingungen, für die man angeben kann, ob sie zwischen zwei Punkten im Raum gültig sind, betrachtet. Diese Nebenbedingungen werden als *spatiale Nebenbedingungen* zusammengefasst.
Ziel der Arbeit ist es ein Planungsverfahren zu entwickeln, welches mit den vom Benutzer gegebenen Zielpunkten $Z = \{z_1, z_2, \ldots, z_n\}$ offline folgende Ergebnisse erstellt:

- Berechnen einer Zielkonfiguration des MRS, bei dem auf jedem $z_i \in Z$ ein Roboter steht und zudem weitere Roboter in der Umgebung als Relaisstationen so platziert sind, dass die spatiale Nebenbedingung eingehalten wird.Diese Zielkonfiguration wird Endkonfiguration S_{end} genannt.

- Berechnen eines Pfades, auf dem die Roboter vom stationären Leitstand aus zu den Positionen aus S_{end} fahren können, ohne die Nebenbedingung zu verletzten.

Dadurch ist der Benutzer mit diesem Planungsverfahren in der Lage, gezielt mehrere Roboter gleichzeitig zu vorgegeben Zielen zu steuern, ohne dabei eine definierte Nebenbedingung zu verletzten.
Dieses Szenario unterscheidet sich grundlegend von den in der Literatur betrachteten Anwendungen, da hier weder eine Umgebung exploriert werden soll, noch das Robotersystem selber eine Abdeckung der Umgebung erreichen soll („Coverage Problem“).

1.2.2 Repräsentation der Umgebung

In der Darstellung der verwandten Arbeiten in Kapitel 2 wird deutlich, dass Planungsverfahren, die Nebenbedingungen für Mehrrobotersysteme berücksichtigen, üblicherweise reaktiv und online arbeiten. Dies führt dazu, dass bestimmte Nebenbedingungen nicht berücksichtigt werden können und „Recovery Strategien“, also die Wiederherstellung der Nebenbedingung nach ihrer Verletzung, implementiert werden müssen.
In dieser Arbeit wird eine Offline-Planung für das Mehrrobotersystem entwickelt. Damit kann eine größere Gruppe von Nebenbedingungen betrachtet werden und der Benutzer ist, schon bevor sich das MRS in Bewegung setzt, darüber informiert, welcher Weg eingeschlagen wird.
In der Planungsphase müssen zwei verschiedene Informationen betrachtet werden: wo ein Roboter fahren kann und zwischen welchen Punkten im Raum die gegebene Nebenbedingung gilt. Diese beiden Informationen werden jeweils als Graph für eine ausgewählte Punktmenge im Raum dargestellt: als *Bewegungsgraph* und als *Bedingungsgraph*. Diese Graphendarstellung führt zu einer Diskretisierung der Umgebung und beeinflusst im Weiteren die gesamte Arbeit. Der Vorteil dieses Ansatzes liegt in seiner Flexibilität. So können mit diesen Graphen sowohl unterschiedliche Roboter als auch eine große Gruppe von Nebenbedingungen modelliert werden.

1.2.3 Nutzung von globalen Mehrroboterplänen

Innerhalb dieser Arbeite wird der Navigationsplan als Offline-Plan erzeugt. Infolgedessen musste eine Darstellungsform gewählt werden, die diesen Plan repräsentiert und die an das Mehrrobotersystem übergeben werden kann. Üblicherweise wird ein Plan für Robotersysteme dadurch erstellt, dass einzelne Roboteraktionen so aneinandergereiht werden, dass ein Übergang vom Startzustand in den gewünschten Endzustand erreicht wird. Da aber das Problem der koordinierten Navigation unter Nebenbedingungen zunächst keine Roboteraktionen betrachtet, sondern nach einer Positionierung in der Umgebung fragt, wurde eine andere Herangehensweise gewählt. Diese Arbeit führt daher die sogenannten *globalen Mehrroboterpläne* als weitere Abstraktionsebene in der Planung ein.

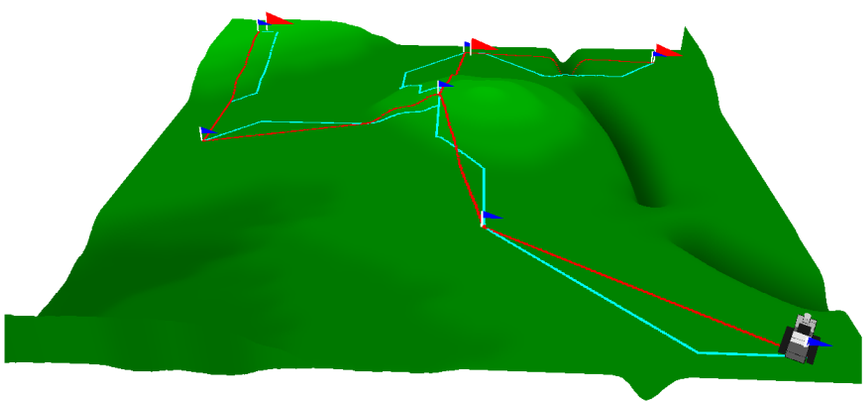


Abbildung 1.1: Beispiel eines globalen Mehrroboterplans in einer simulierten Umgebung. Der Roboter markiert den Startpunkt des MRS. Die roten Fahnen sind vom Benutzer ausgewählte Zielpunkte, die blauen Fahnen vom System berechnete zusätzliche Wegpunkte. In Türkis sind die Pfade eingezeichnet, die die Roboter nehmen sollen.

Diese Abstraktionsebene hat den Vorteil, dass auf ihr die Positionsplanung und die Pfadplanung einfacher durchgeführt werden können. Dabei besteht ein globaler Mehrroboterplan aus Zielpunkten, auf denen je ein Roboter stehen soll, sowie der Nachbarschaftsbeziehung der Zielpunkte, die definiert, in welcher Reihenfolge die Punkte angefahren werden müssen. Zudem werden die vorausberechneten Pfade, auf denen sich die Roboter bewegen müssen, eingetragen. Ein Beispiel ist in Abbildung 1.1 zu sehen.

1.3 Wissenschaftlicher Beitrag

Die vorliegende Arbeit beschäftigt sich mit der koordinierten Navigation eines Mehrrobotersystems unter Nebenbedingungen. Wie in den meisten Mehrroboteranwendungen ist es auch hier notwendig, dass die Roboter kooperieren, um den Aktionsradius des Gesamtsystems zu erhöhen. Entstanden ist ein zentrales Offline-Planungsverfahren, welches es einem einzelnen Benutzer ermöglicht, das Mehrrobotersystem gezielt einzu-

setzen.
Im Rahmen der erstmaligen Betrachtung des neuen Problems der koordinierten Navigation unter spatialen Nebenbedingungen sind folgende Methoden, Werkzeuge und Ergebnisse gefunden worden:

- Koordinierte Navigation eines Mehrrobotersystems unter spatialen Nebenbedingungen:
  Während das Coverage-Problem (Einführung des Coverage-Problem siehe Gage [1994]), bei dem die Roboter selbstständig eine meist unbekannte Umgebung möglichst gut abdecken sollen, ein gut untersuchtes Problem darstellt (z.B. Batalin and Sukhatme [2002] oder Howard et al. [2002]), ist diese Variante, bei der der Benutzer die Positionen, an denen die Roboter benötigt werden, festlegt, bisher nicht betrachtet worden. Diese Arbeit definiert dieses Problem, präsentiert und analysiert einige Eigenschaften des Problems und schlägt verschiedene Lösungswege vor.
- Äquivalenz zwischen dem Steinerbaum-Problem und der Endpositionierung:
  In der gewählten Repräsentation der Umgebung konnte gezeigt werden, dass das Finden einer Positionierung der Roboter, die die Nebenbedingung nicht verletzt, dem Steinerbaum-Problem entspricht. Dies ermöglichte es, eine Lösung mit Hilfe bekannter Steinerbaum-Heuristiken (z.B. Mehlhorn [1988]) zu finden, die als Benchmark für weitere Lösungen genutzt werden kann, da das Steinerbaum Problem sehr gut untersucht ist. Zudem zeigt dieser Zusammenhang, dass eine optimale Lösung voraussichtlich nicht in einem sinnvollen Zeitansatz gefunden werden kann.
- Kombination von Pfad- und Nebenbedingungsinformation:
  Die Informationen, wie ein Roboter sich bewegen kann und wie er sich (nach der Nebenbedingung) bewegen darf, wird in zwei separaten Graphen dargestellt. Mit Hilfe der vorgestellten „Seperated Connection Graphs“ (SCG) werden diese grundsätzlich unterschiedlichen Informationen verbunden und es entstehen Teilgraphen, die zusammengesetzt eine Lösung für das Navigationsproblem unter Nebenbedingung darstellen.
- Globale Mehrroboterpläne:
  Während die Lösungen, die nicht reaktiv ein Mehrrobotersystem steuern, üblicherweise auf Roboteraktionen planen (siehe z.B. Rooker and Birk [2007]), wird in dieser Arbeit eine zusätzliche Abstraktionsschicht eingeführt. Diese globalen Mehrroboterpläne können mit den vorgestellten Algorithmen schnell erstellt werden und bilden die Basis für die Steuerung des Mehrrobotersystems. Dabei ist die Steuerung von der Planung abgekoppelt und die Quelle eines globalen Mehrroboterplans kann beliebig sein. Zudem bieten die globalen Mehrroboterpläne eine gute Möglichkeit, die geplanten Bewegungen der Roboter zu visualisieren.
- Planer zur Erstellung des globalen Mehrroboterplans:
  In der Arbeit werden zwei Verfahren zur Erstellung eines globalen Mehrrobo-

terplans vorgestellt. Der STPlaner basiert auf dem Melhorn-Algorithmus (Mehlhorn [1988]) und ist daher als „Baseline“ sehr gut geeignet. Der STPlaner erstellt Pläne, die besonders wenig Roboter in der Endkonfiguration benötigen. Der zweite Planungsalgorithmus, genannt AgentPlaner, ist ist dagegen insbesondere für die praktische Anwendung geeignet. Der auf dem MPR-Flooding Algorithmus (siehe Tc-REs [1995] und [Qayyum et al., 2002]) basierende Planer berechnet Pläne, die insbesondere eine schnelle Ausführungszeit haben. Zudem ist er auch für dynamische oder unbekannte Umgebungen geeignet, da er bekannte Pläne bei veränderten Umgebungen oder Nebenbedingungen online anpasst.

- Erzeugung von Handlungsanweisung aus globalen Mehrroboterplänen:
  Um ein Mehrrobotersystem mit dem gefundenen globalen Mehrroboterplan zu steuern, müssen dem einzelnen Roboter Handlungsanweisungen gegeben werden. Hierzu wurde ein Satz von Befehlen entwickelt, die auf den Grundzügen „Bewegung“ und „Warten“ basieren. Da der Zeitpunkt, wann eine Bewegung ausgeführt werden darf, davon abhängt, wann ein anderer Roboter sein Ziel erreicht, werden diese Bedingungen in die Handlungsanweisungen kodiert.
- Benötigte Anzahl von Robotern in einem globalen Mehrroboterplan:
  Bei der Übertragung der globalen Mehrroboterpläne auf die Handlungsanweisungen stellte sich das Problem, dass die Anzahl der benötigten Roboter für einen globalen Mehrroboterplan nicht bekannt ist. Die Lösung dieses Problems stellte eine besondere Form der Baumtraversierung mit Agenten dar (z.B. Averbakh and Berman [1996] und Pierre Fraigniaud et al. [2006]). Hierzu konnte in Zusammenarbeit mit der Abteilung I der Informatik der Universität Bonn ein optimaler Algorithmus gefunden werden. Zudem wurde in weiterführenden Arbeiten von Herrn Dr. Elmar Langetepe und Herrn Andreas Lenerz gezeigt, dass dieses Traversierungsproblem auf allgemeinen Graphen NP-vollständig ist.
- Online Verbesserung der Ausführungsgeschwindigkeit:
  Da die Handlungsanweisungen unter bestimmten Bedingungen zwischen den Robotern getauscht werden können, können Optimierungen bei der Planausführung einfach realisiert werden. So kann mit einem einfachen Mechanismus z.B. den schnellen Robotern die langen Handlungsanweisungen zugeteilt werden und den langsamen Robotern die kurzen. Diese Zuteilung erfolgt automatisch, ohne dass vorher die Geschwindigkeit der Roboter bekannt sein muss. Auch eventuelle Deadlocks (siehe Rausch and Levi [1996]), die durch enge Stellen in der Umgebung entstehen, können einfach durch das Tauschen von Handlungsanweisungen aufgelöst werden.

1.4 Veröffentlichungen

Teile dieser Dissertation sind auf Konferenzen, in Tagungsbänden und in einer Fachzeitschrift veröffentlicht worden. Dies sind im einzelnen:

- Bernd Brüggemann, Dirk Schulz *Coordinated navigation for multi-robot systems with additional constraints (extended abstract)* in Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 2010 (Brüggemann and Schulz [2010a])
- Bernd Brüggemann, Dirk Schulz *Coordinated navigation of multi-robot systems with binary constraints* in Proceeding of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010 (Brüggemann and Schulz [2010b])
- Bernd Brüggemann, Michael Brunner, Dirk Schulz *Spatially constrained coordinated navigation for a multi-robot system* in Ad Hoc Networks, Volume 11, Issue 7, September 2013, Pages 1919-1930, ISSN 1570-870 (Brüggemann et al. [2013a])
- Bernd Brüggemann, Elmar Langetepe, Andreas Lenerz, Dirk Schulz *From a Multi Robot Global Plan to Single Robot Actions* in Proceedings of the International Conference on Informatics in Control, Automation and Robotics (ICINCO), 2012 (Brüggemann et al. [2012])
- Bernd Brüggemann, Michael Brunner, Dirk Schulz *Outdoor Navigation with a Coordinated Multi-Robot System that maintains Spatial Constraints* in Proceeding of 2nd IFAC Workshop on Multivehicle Systems, 2012 (Brüggemann et al. [2012])
- Bernd Brüggemann, Michael Brunner, Dirk Schulz *Asynchronous Flooding Planner for Multi-Robot Navigation* in Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics (ICINCO), 2013 (Brüggemann et al. [2013])
- Bernd Brüggemann, Elmar Langetepe, Andreas Lenerz *A strategic occupation game on graphs* in Proceedings of Middle-European Conference on Applied Theoretical Computer Science (MATCOS), 2013 (Brüggemann et al. [2013b])

1.5 Struktur der Arbeit

Nach dieser Einleitung werden in Kapitel 2 andere Arbeiten im Bereich der koordinierten Mehrrobotersteuerung vorgestellt, und die Ergebnisse dieser Arbeit werden eingeordnet. Kapitel 3 zeigt die Grundlagen aus dem Bereich der algorithmischen Geometrie und der Kommunikationstechnik auf, auf denen die neuen Erkenntnisse aufbauen. Mit Kapitel 4 beginnt die Arbeit mit der Betrachtung der globalen Planung und der Erstellung von globalen Mehrroboterplänen. Zunächst werden für diese globale Planung die notwendigen Datenstrukturen und Annahmen vorgestellt (Kapitel 4.1). Daraufhin werden Seperated Connection Graphs (SCG) eingeführt und erläutert. Diese Struktur ermöglicht eine schnelle Planung auf dem Bewegungs- und Bedingungsgraph. Die Berechnung der Zielkonfigurationen des MRS wird in Kapitel 4.2 beschrieben. Hier werden die zwei zentralen Algorithmen STPlaner und AgentPlaner entwickelt und deren

Eigenschaften gezeigt. In Kapitel 4.3 werden die Wege zu den gefundenen Zielpunkten berechnet. Allen Unterkapiteln gemein ist, dass die Planung jeweils unter Beachtung der Nebenbedingung geschieht. In Kapitel 4.4 werden dann einige Beispielplanungen in der Simulation gezeigt. Schon diese Beispiele zeigen, dass die vorgestellten Algorithmen ganz unterschiedliche Eigenschaften besitzen.
Ein Vergleich der vorgestellten Algorithmen STPlaner und AgentPlaner zusammen mit der Variante FastAgentPlaner wird in Kapitel 5 vorgenommen. Hierbei dient im Leistungsvergleich der STPlaner als Messbasis, da der zugrunde liegende Algorithmus theoretisch gut erfasst ist. So kann gezeigt werden, dass jeder der drei Ansätze in bestimmten Bereichen seine Stärken hat und somit je nach Anforderung besonders geeignet ist.
Bis zu diesem Zeitpunkt ist bei der Planung davon ausgegangen worden, dass die gesamte Umgebung vorher bekannt ist und sich auch während der Ausführung nicht ändert. Diese Annahme ermöglichte die Entwicklung der beschriebenen Algorithmen. Um auch einen praktischen Nutzen aus diesen Überlegungen gewinnen zu können, müssen jedoch auch veränderliche Umgebungen betrachtet werden. Dies geschieht in Kapitel 6. So wird der AgentPlaner dafür genutzt, in Umgebungen, in denen zusätzliche Hindernisse auftauchen können, zu planen (Kapitel 6.1). Neben der Planung auf veränderlichen Umgebungen wird auch die Planung in unbekannten Umgebungen mit dem AgentPlaner betrachtet (Kapitel 6.2). Hier wird gezeigt, dass der AgentPlaner dazu geeignet ist, unbekannte Umgebungen zu behandeln, jedoch aufgrund von instabilen Plänen die Ausführungszeit sehr hoch sein kann. Um die Änderungen des globalen Plans bei veränderten Umgebungen zu berücksichtigen, müssen auch die Handlungsanweisungen der einzelnen Roboter geändert werden. Wie alte Handlungsanweisungen in neue überführt werden können, wird in Kapitel 6.3 besprochen.
Nach der Betrachtung der Simulationen und der Möglichkeiten, auf dynamische Umgebungen zu reagieren, wird in Kapitel 7 die Übertragung auf reale Systeme betrachtet. Dazu werden zunächst die Änderungen, die sich durch den Einsatz eines realen Robotersystems gegenüber der Simulation ergeben, diskutiert sowie das, für verschiedene reale Versuche genutzte Mehrrobotersyste, beschrieben. Einige dieser Versuche werden in Kapitel herausgegriffen und detailliert beschrieben.
Die Arbeit endet mit einer Diskussion der erreichten Ergebnisse und mit einem Ausblick auf mögliche weiterführende Arbeiten (Kapitel 8).

2 Verwandte Arbeiten

In diesem Kapitel wird die vorliegende Arbeit in den Themenkomplex der koordinierten Mehrrobotersysteme eingeordnet. Dazu werden zunächst bekannte Systematiken vorgestellt, die einen Einblick geben, welche Arten der Koordination und Kooperation von Robotern existieren. Nach der Einordnung der vorliegenden Arbeit in solche Systematiken werden verschiedene Arbeiten aus der Literatur vorgestellt, die die unterschiedlichen Ansätze umsetzen. Insbesondere werden dabei Arbeiten besprochen, die neben der Koordination auch den Aspekt von Nebenbedingungen betrachten.

2.1 Bewegungsplanung in Mehrrobotersystemen

In Latombe [1991] wird die Bewegungsplanung für Roboter als „How can a robot decide what motions to perform in order to achieve goal arrangements of physical objects“ (Latombe [1991], Preface Seite ix, 5. Auflage) beschrieben. Das bedeutet, die Bewegungsplanung beschäftigt sich mit den expliziten Wegen eines Roboters in seiner Umwelt. Allerdings stellt Latombe ebenfalls fest: „The elementary operative intelligence that people use unconsciously to interact with their environment [...] turns out to be extremely difficult to duplicate using a computer-controlled robot“ (Latombe [1991], Preface Seite x, 5. Auflage), also dass die normalen Interaktionen eines Menschen mit der Umwelt (z.B. Wegplanung, Hindernisvermeidung) für einen Roboter schwierig nachzuvollziehen sind. Gerade die Interaktion mit der Umwelt ist es, die die schwierigsten, aber auch interessantesten Probleme für die Robotik aufwirft. Es sei schon hier darauf hingewiesen, dass die Interaktion mit der Umwelt auch die Interaktion mit anderen Robotern einschließt.
Dass die Bewegungsplanung für Roboter sich aber nicht alleine auf die Hindernisver-

meidung bezieht, stellt Latombe im gleichen Abschnitt klar und auch sein Buch geht weit über dieses Themengebiet hinaus: Es beinhaltet unter anderem die Berechnung von kollisionsfreien Wegen mit beweglichen Hindernissen, Multi-Robot-Koordination, Bewegen von Objekten, Umgang mit Unsicherheiten, physikalische Weltmodelle sowie stabiles Greifen von Objekten (vergleiche Latombe [1991]).
Die vorliegende Arbeit befasst sich mit der globalen Bewegungsplanung von Mehrrobotersystemen. Das heißt, es soll für mehrere Roboter ein Plan erstellt werden, wie sie in ihrer Umgebung bewegt werden können, um ein bestimmtes Ziel zu erreichen. Zusätzlich zu den Problemen, die die Planung eines einzelnen Roboters in der Umgebung erzeugt, entstehen dabei neue Probleme; es ergeben sich jedoch auch neue Möglichkeiten.
Bewegen sich mehrere Roboter in der gleichen Umgebung, beeinflussen sie sich gegenseitig. Diese Beeinflussung kann negativ sein, wie z.B. das gegenseitige Blockieren von Wegen. Die Roboter können sich jedoch auch unterstützen, in dem sie z.B. als Relaisstationen die Funkreichweite erweitern. Solche Effekte müssen bei der globalen Mehrroboterplanung berücksichtigt werden.
Auch wenn die Betrachtung von Agentensystemen und ihre Bewegungen schon älter sind, wird die Bewegungsplanung eines Mehrrobotersystems erst ab 1990 eingehender erforscht. Eine der ersten Arbeiten, die den damals aktuellen Stand der Bewegungsplanung für Mehrrobotersysteme systematisierte, ist Cao et al. [1995]. Dabei fassten die Autoren zunächst jene Gründe zusammen, warum ein Mehrrobotersystem überhaupt eingesetzt werden soll:

- Die Aufgabe ist zu komplex, als dass ein Roboter alleine sie lösen kann, oder der Einsatz von mehreren Robotern führt zu einer deutlich erhöhten Performance.
- Viele einfache Roboter sind billiger herzustellen, flexibler und deutlich ausfallsicherer.
- Die Betrachtung, wie ein System aus Robotern eine Aufgabe löst, kann Rückschlüsse auf soziales Veralten innerhalb von biologischen Systemen gewähren.

Cao et al. konnten in der Literatur, für sie überraschend, nur drei verschiedene Definitionen von Kooperationen zwischen Robotern finden. Diese Definitionen sind nach Cao et al. [1995] wie folgt:

- „Joint collaborative behavior that is directed toward some goal in which there is a common interest or reward.“ (nach Barnes and Gray [1991])
  Hierbei geht es also um das Erreichen eines gemeinsamen Zieles. Nach Cao et al. führt diese Definition insbesondere in solche Aufgabengebiete wie der „Task Allocation“, der „Task Decomposition“ sowie weitere Bereiche der verteilten Intelligenz.
- „A form of interaction, usually based on communication.“ (nach Mataric [1994])
  Diese Definition impliziert nach Cao et al. den Themenbereich der Kommunikation mit der damit verwandten Ressourcenverwaltung und der Ausfallsicherheit.

- „[Joining] together for doing something that creates a progressive result such as increasing performance or saving time.“ (nach Premvuti and Yuta [1990])
  Cao et al. sehen hierin eine Definition, die auf die Messbarkeit der Vorteile von Kooperationen abzielt.

Zu den gefundenen Definitionen der Kooperation zwischen Robotern fügen Cao et al. noch eine eigene hinzu: „Given some task specified by a designer, a multiple-robot system displays cooperative behaviour if, due to some underlying mechanism [. . .], there is an increase in the total utility of the system.“ (vergleiche Seite 2, vorletzter Abschnitt in Cao et al. [1995]). Diese Definition ist möglichst allgemein gehalten und spezifiziert den „Kooperationsmechanismus“ nicht weiter. Es wird nur verlangt, dass die Nutzbarkeit des Mehrrobotersystems durch die Kooperation gegenüber einem System aus mehreren Robotern, die nicht kooperieren, steigt. Zusätzlich zu einer Definition von Kooperation definieren Cao et al. fünf Richtungen („research axis“), mit denen die Kooperation in einem Mehrrobotersystem vorangetrieben werden kann:

- „Group architecture“: Die Soft- und Hardware-Architektur, die es den Robotern ermöglicht, zu kooperieren.
- „Resource conflicts“: Der Umgang mit gemeinsamen Ressourcen.
- „Origins of cooperation“: Wie entsteht und wie erreicht man Kooperation?
- „Learning“: Eine der Schlüsselfähigkeiten, um flexibel auf verschiedene Situationen und Aufgaben zu reagieren.
- „Geometric problems“: Umfasst Probleme die, wie die Mehrroboterpfadplanung, geometrisch ausgedrückt und z.B. über Graphen gelöst werden können.

Nach der Definition aus Cao et al. [1995] kooperieren die Roboter in dem in dieser Arbeit vorgestellten Planungsframework. Dazu müssen zwei Bedingungen erfüllt sein: es muss eine Steigerung des Nutzens sichtbar sein und ein expliziter Mechanismus muss die Koordination ermöglichen. Dabei liegt der Nutzen darin, dass die Roboter Bereiche erreichen können, die von einzelnen unkoordinierten Robotern nicht erreicht werden können, wenn die Nebenbedingung nicht verletzt werden darf. Dies wird durch eine gemeinsame globale Planung erreicht. Innerhalb der „research axis“ wird in der vorliegenden Arbeit die Richtung „Resource conflicts“ und stärker die „Geometric problems“ betrachtet, um das Problem zu lösen.
Etwa zehn Jahre später stellten Arai et al. [2002] den neuen Stand der Bewegungsplanung für Mehrrobotersysteme vor. Sie ordnen die Fortschritte in der Koordination von Mehrrobotersystemen verschiedenen Themengebieten („principle topics“) zu:

- Biologisch inspiriert
- Kommunikation
- Architekturen, Aufgabenzuweisung und Exploration
- Transport von Objekten
- Bewegungskoordination

- Veränderliche Roboter („reconfigurable robots“)

Dabei bescheinigen Arai et al., dass in allen diesen Bereichen in den vergangenen zehn Jahren deutliche Fortschritte erreicht wurden. Insbesondere heben sie den Bereich Kommunikation und die Gebiete um die Architekturen hervor; hier seien die größten Fortschritte erzielt worden. Im Vergleich dazu sehen sie die veränderlichen Roboter nicht ganz so stark vertreten. Zuletzt stellen Arai et al. noch sechs Fragen, die, ihrer Meinung nach, unter vielen anderen noch ungelöst sind. Besonders zu erwähnen ist hier die zweite Frage: „How do we easily enable humans to control multi-robot teams?“ (in Arai et al. [2002], Seite 659). Die hier vorliegende Arbeit gibt hierauf eine Teilantwort, da mit Hilfe der globalen Mehrroboterpläne die Intention des MRS leichter zu verstehen ist und mit der Planungskomponente ein Mehrrobotersystem gezielt von einem einzelnen Operateur gesteuert werden kann.

2.1.1 Systematiken

Neben der grundsätzlichen Klassifikation von Forschungsthemen in der Koordination von Robotern gibt es verschiedene Ansätze die konkreten Lösungen zu bestimmten Problemen zu klassifizieren. Eine der ersten Systematiken wurde von Dudek et al. [1996] vorgeschlagen.

Charakterisierung nach Dudek et al. [1996]

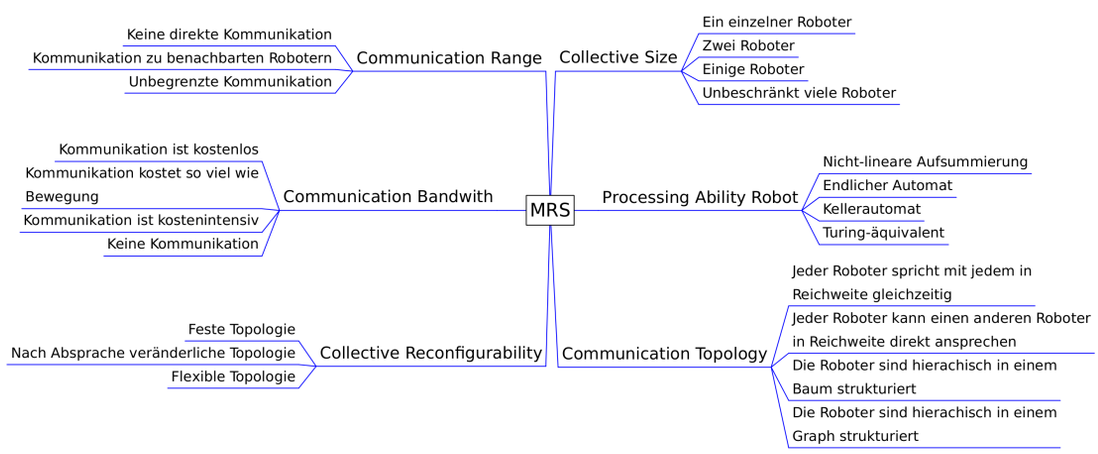


Abbildung 2.1: Systematik nach Dudek et al. [1996]

Dudek et al. definieren dabei 6 verschiedene Eigenschaften, die mit fest zugeordneten Attributen versehen werden. Die Gruppierung dieser Eigenschaften und ihrer Attribute

ist in Abbildung 2.1 zu sehen. Für jede Eigenschaft gibt es dabei drei bis vier Kennzeichner. Detailliert messen die Eigenschaften:

- „Collective Size“: Die Anzahl der verwendeten Roboter.
- „Communication Range“: Die Reichweite der Kommunikation der Roboter untereinander.
- „Communication Topology“: Kommunikation kann nicht nur durch die Reichweite eingeschränkt sein, sondern auch z.B. durch Hierarchien.
- „Communication Bandwidth“: Die Kosten, die für die Kommunikation entstehen.
- „Collective Reconfigurability“: Die Fähigkeit des Mehrrobotersystems, seine Konfiguration bzw. Topologie zu verändern.
- „Processing Ability of Each Collective Unit“: Die Komplexität der Berechnungen, die der einzelne Roboter durchführen kann.

Diese Systematik ermöglicht es verschiedene Ansätze getrennt von ihrer Applikation zu klassifizieren. In Dudek et al. [1996] werden einige bekannte Arbeiten aus den 90er Jahren so eingeteilt. Der hier vorliegende Ansatz zur koordinierten Navigation unter Nebenbedingungen kann wie folgt klassifiziert werden:

- *Einige Roboter*
  Ein Ziel ist es, die Zielpunkte mit möglichst wenigen Robotern zu erreichen. Zwei Roboter sind jedoch im allgemeinen Fall zu wenige.
- *Kommunikation zu anderen Robotern in der Nähe*
  Da immer ein reales Robotersystem als Grundlage für die Betrachtung des MRS in dieser Arbeit dient, wird eine begrentze Kommunikation angenommen. Auch wärend der Plansuführung werden dabei weiterhin Informationen zwischen den Robotern ausgetauscht.
- *Jeder Roboter kann einen anderen Roboter in Reichweite direkt adressieren*
  Die Roboter sind in der Lage, sich über eine eindeutige ID gegenseitig direkt zu adressieren.
- *Kommunikation ist kostenlos*
  Sowohl unter Energie-, als auch unter Zeitaspekten fällt die Kommunikation in dieser Anwendung nicht ins Gewicht.
- *Die Topologie wird nach Absprache der Roboter geändert*
  Die Einhaltung der Topologie innerhalb des MRS ist ein wichtiger Punkt, da sie garantiert, dass die Nebenbedingung eingehalten wird. Allerdings kann die Topologie z.B. in dynamischen Umgebungen neuen Erfordernissen angepasst werden.
- *Turing-äquivalent*
  Die Roboter müssen komplexe Systeme darstellen, da die Planung Probleme der lokalen Navigation nicht berücksichtigt, dies von den Robotern also als Funktion erwartet wird.

Charakterisierung nach Todt et al. [2000]

Eine Klassifikation nur für Navigationsansätze zeigen Todt et al. [2000] auf. Dabei werden nur Koordinationsverfahren klassifiziert, die die Navigation eines Mehrrobotersystems behandeln. Insbesondere in solchen Fällen, in denen die Roboter einen gemeinsamen Arbeitsraum haben, ist die Koordination der Bewegungen notwendig, um Deadlocks zu vermeiden und die Ausführungszeit zu minimieren. Ohne eine solche Koordination stellen die anderen Roboter immer dynamische Hindernisse dar. Auch wenn dies nicht der Hauptaspekt dieser Arbeit ist, kann die koordinierte Navigation unter Nebenbedingung auch hiermit klassifiziert werden:

- Koordinierungsmethode: Gekoppelt
  Die Koordination ist zentral angelegt und die Pfade für die Roboter werden gleichzeitig erzeugt.
- Koordinierungszeit: Variabel und offline
  Der globale Mehrroboterplan wird zu Beginn erzeugt. Wenn sich die Umgebung in Laufe der Zeit ändert, kann darauf reagiert werden.
- Prioritäten: Ohne Prioritäten
  Den einzelnen Robotern werden keine unterschiedlichen Prioritäten zugeordnet.
- Zu optimierender Messwert: Kürzeste Ausführungszeit
  Ziel der Planung ist eigentlich, die Anzahl benötigter Roboter zu minimieren. Solch eine Metrik ist aber bei Todt et al. [2000] nicht vorgesehen. Ist die Anzahl benötigter Roboter aber berechnet, werden die Weglängen minimiert und damit die Ausführungszeit reduziert.
- Arbeitsbereich: physical space
  Die gesamte Planung findet auf einem Modell der Umgebung statt.

Charakterisierung nach Iocchi et al. [2001]

Iocchi et al. [2001] betrachten bei ihrem Ansatz einer Klassifikation von Mehrrobotersystemen die Art der Koordinierungsmethode. Dabei werden zunächst Mehrrobotersysteme, die kooperieren, von denen unterschieden, bei denen die Roboter sich zwar im gleichen Arbeitsraum befinden, aber nur nebeneinander arbeiten. Kooperierende Systeme werden dann wiederum unterteilt in Systeme, bei denen die Roboter explizit voneinander wissen, und solche, bei denen die Roboter nur implizit z.B. durch Änderungen in der Umgebung voneinander Notiz nehmen können. Auch hier wird wieder nur die stärkere Koordinierung weiter aufgeschlüsselt. Dabei handelt es sich um koordinierte Mehrrobotersysteme, in denen die einzelnen Roboter Kenntnisse über den Status der anderen Roboter haben. Bei solchen Systemen wird zwischen einer starken und einer schwachen Form der Koordination unterschieden. Dabei hat die starke Form ein explizites Protokoll zur Koordination. In der starken Form der Koordination unterscheiden Iocchi et al. [2001] zwischen zentralisierten und dezentralisierten Verfahren und bei den

zentralisierten Verfahren nochmals zwischen stark und schwach zentralisierten Verfahren. Diese Systematik ist in Abbildung 2.2 zu sehen.

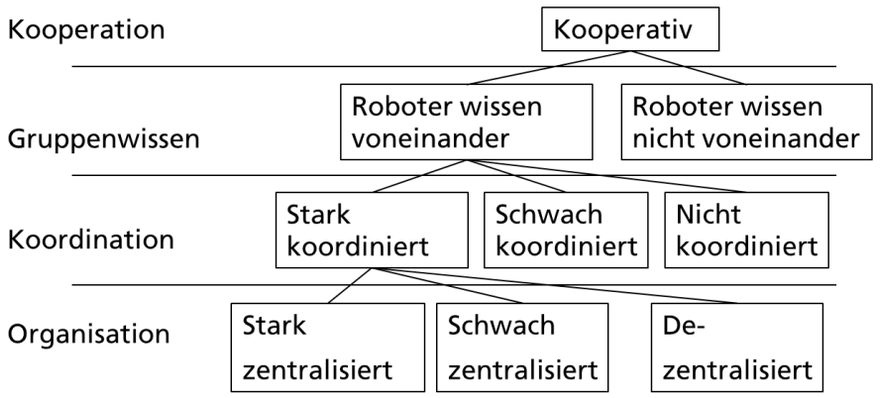


Abbildung 2.2: Systematik der Koordination in Mehrrobotersystemen nach Iocchi et al. [2001]. Die jeweils am stärksten koordinierte Lösung wird weiter unterteilt. Die koordinierte Navigation unter Nebenbedingung in der vorliegenden Arbeit ordnet sich in den stark zentralisierten Verfahren ein.

Anhand dieser sehr unterschiedlichen Systematiken für die Koordination von Mehrrobotersystemen kann die Bandbreite der Themen in diesem Forschungsfeld ermessen werden. Zudem gibt es noch viele weitere Systematiken, die sich mit speziellen Feldern der Koordination beschäftigen, wie z.B. Parker [2008] das Feld der Verteilten Intelligenz beschreibt und systematisiert. Diese Systematiken zeigen auch, dass es ganz unterschiedliche Verfahrensweisen gibt, die eine Koordination von mehreren Robotern ermöglichen, und dass der hier vorgestellte Ansätze zur koordinierten Navigation eines Mehrrobotersystems unter spatialen Nebenbedingungen nur eine Möglichkeit zur Lösung des gestellten Problems darstellt.

2.2 Kooperation und Koordination in Mehrrobotersystemen

Wie im vorigen Kapitel beschrieben, existieren vielfältige Ansätze Roboter innerhalb eines Mehrrobotersystems miteinander kooperieren zu lassen und sie zu koordinieren. Dabei sind die meisten der Ansätze anwendungsgetrieben, d.h. eine bestimmte Aufgabe soll mit einem Mehrrobotersystem bewältigt werden. Nur vergleichsweise wenige Arbeiten beschäftigen sich mit einem Ansatz, der grundsätzliche Fähigkeiten der Koordination zur Verfügung stellt. So stellen z.B. Fierro et al. [2002] ein Roboterframework vor, das eine

Koordination ermöglicht, die dann für das gesamte Aufgabenspektrum eingesetzt werden kann; Kaminka et al. [2010] betrachten dieses Problem aus der spiel-theoretischen Sicht. Kalech et al. [2006] beschreiben eine allgemeine Möglichkeit, Fehler in der geplant Koordinierung zu finden. So kann z.B. festgestellt werden, dass einzelne Roboter den Status der anderen Roboter falsch einschätzen und damit unerwünschte Verhaltensweisen zeigen.
Typisch für Mehrroboteraufgaben sind die koordinierte Navigation, kooperative Lokalisierung, die Aufgabenverteilung, die Exploration und das sogenannte Coverage Problem. Da jede dieser Aufgaben mit ganz unterschiedlichen Ansätzen gelöst werden kann, werden diese Aufgaben und typische Lösungen hier vorgestellt.

2.2.1 Koordinierte Navigation

Als ein wichtiges Thema in der Koordination eines Mehrrobotersystems wird die Koordination der Bewegung der einzelnen Roboter betrachtet. Werden hier die Systeme nicht koordiniert, bilden die einzelnen Roboter dynamische Hindernisse füreinander.
Dabei ist gerade in engen oder mit vielen Hindernissen gefüllten Umgebungen eine Koordination wichtig. Hier können schnell Situationen entstehen, in denen die Roboter sich gegenseitig behindern. Ein übliches Verfahren, die Wege zu planen, ist eine Priorisierung der Roboter. Dabei werden die Wege für den Roboter mit der höchsten Priorität zuerst geplant und mit diesem Wissen dann, sortiert nach Priorität, die der anderen Roboter (siehe Buckley [1989]; Clark et al. [2003]). Dies bedingt jedoch eine zentrale Koordinierung. Aber auch dezentrale Lösungen sind möglich, bei denen die Roboter online ihre Pfade anpassen (z.B. in Svestka and Overmars [1998]; Velagapudi et al. [2010]) oder die Geschwindigkeiten so anpassen, dass keine Kollisionen oder Behinderungen entstehen (z.B. in Simeon et al. [2002]; van den Berg and Overmars [2005] und Cui et al. [2012]). Müssen die Roboter sich durch sehr beengte Umgebungen, wie z.B. eine Büroumgebung mit engen Fluren und Türen, bewegen, entstehen oft sogenannte Deadlock-Situationen (beschrieben z.B. in Rausch and Levi [1996]). Die Roboter müssen in einer bestimmten Reihenfolge bewegt werden, um anderen Robotern den Weg frei zu machen. So kann es auch vorkommen, dass einzelne Roboter einen größeren Umweg in Kauf nehmen müssen. Hier bieten z.B. Bennewitz et al. [2001] Optimierungsmethoden an, um die gesamte Ausführungszeit möglichst kurz zu halten.
Aber nicht nur das Erreichen von Zielen der einzelnen Roboter kann durch Kooperation verbessert oder erst ermöglicht werden, auch andere Aufgaben basieren auf der Koordination von Bewegungen. Bei sogenannten „Leader-Follower“ Ansätzen führen ein oder mehrere Roboter andere Roboter zu einem Ziel. So bildet das Mehrrobotersystem eine Art Konvoi, bei dem nur der Führungsroboter die globale Pfadplanung durchführen muss, während die folgenden Roboter nur noch lokal planen müssen (z.B. in Hoeller et al. [2008]). Dies kann zum einen dazu genutzt werden, viele kleine Roboter, die nur eine begrenzte Sensorik besitzen, von einigen großen Robotern zu einem Ziel zu gelei-

ten (in Parker et al. [2004]). So erreicht man, dass nicht alle Roboter in der Gruppe mit aufwändiger (und teurer) Sensorik ausgestattet werden müssen. Zum anderen kann ein „Leader-Follower“-System auch dafür eingesetzt werden, die Reichweite des Gesamtsystems zu erhöhen. In den Arbeiten Carpin and Parker [2002] Nguyen et al. [2002, 2003, 2004]; Pezeshkian et al. [2006] zeigen Nguyyen et al. schrittweise, wie die Kommunikation eines Führungsroboters mit einer ortsfesten Basisstation mit Hilfe der Folgeroboter aufrechterhalten werden kann. Dazu überprüft der letzte Folgeroboter ständig die Signalqualität der Funkverbindung. Wird diese zu schlecht, bleibt der letzte Folgeroboter stehen, um eine Relaisstation zu bilden. So verfügt der Führungsroboter über intelligente Relaisstationen, die auch in schwierigen Umgebungen wie z.B. einem Bunker, für eine funktionierende Funkverbindung sorgen.
Das „Leader-Follower“-Verhalten kann als Spezialfall einer Formation angesehen werden. Bei einer Formation versuchen die Roboter ein bestimmtes Muster bzw. bestimmte Abstände zueinander einzuhalten. Dabei werden zentral koordinierte Verfahren wie z.B. in Tanner and Kumar [2005] und Liu et al. [2010] lokalen Verfahren (Urcola and Montano [2009]) gegenübergestellt. Während in den zentral koordinierten Ansätzen vor allem das Verhalten an Hindernissen untersucht wird, stellt sich bei den dezentralen Ansätzen auch das Problem, dass das Verhalten der einzelnen Roboter so konvergiert, dass die gewünschte Formation entsteht. In einer weiteren Form des Leader-Follower-Ansatzes wird der Leader durch einen Menschen repräsentiert. Dies erschwert die Aufgabe, da die Roboter auf die Aktionen eines Menschen reagieren müssen. Ein Beispiel findet sich in Parker and Howard [2009].
Bei einem Einsatz zur Überwachung einer großen Umgebung kann der Vorteil eines Mehrrobotersystems, von verschiedenen Orten gleichzeitig Informationen gewinnen zu können, genutzt werden. Hierzu müssen sie koordiniert patrouillieren, d.h. auf mehr oder weniger fest vordefinierten Wegen regelmäßig alle Punkte des zu überwachenden Gebietes besuchen. Lösungen können über die Betrachtung von Graphen Elmaliach et al. [2007], verhaltensbasierte Ansätze Kowalczyk [2001] und „Leader-Follower“-Systeme erreicht werden Iocchi et al. [2011].

2.2.2 Kooperative Lokalisierung

Lokalisierung, also das Wissen darüber, wo man ist, ist für die meisten Anwendungen in der Robotik essenziell. Dabei sind alle Lokalisierungsmethoden, die genutzt werden können, fehlerbehaftet. Indem mehrere Roboter kooperativ ihre Position bestimmen, kann dieser Fehler verringert werden, wobei die Anzahl der Roboter dabei einen positiven Einfluss auf die Genauigkeit hat (vergleiche Schneider and Wildermuth [2012]). Dabei können die Informationen über die Beobachtungen der einzelnen Roboter entweder gesammelt und zu einer konsistenten Positionsschätzung genutzt werden (in Fox et al. [2000]; Howard et al. [2003]) oder jeder einzelne Roboter erhält die Informationen der anderen (verfügbaren) Roboter und nutzt diese zur Verbesserung der eigenen

Positionsschätzung (in Mu et al. [2011]).

2.2.3 Aufgabenverteilung

Bei der Aufgabenverteilung, der „Task Allocation“ in einem Mehrrobotersystem, wird angenommen, dass die Aufgabe, die dem Mehrrobotersystem gestellt wurde, in mehrere Teilaufgaben unterteilt werden kann. Durch eine geschickte Verteilung dieser Teilaufgaben kann nun die Ausführungsgeschwindigkeit verbessert werden. Dazu muss das Mehrrobotersystem jedoch erkennen können, welche Aufgaben gleichzeitig erfüllt, aber auch welche Aufgaben z.B. nur von einem bestimmten Roboter geleistet werden können. Auch hier liegt der essentielle Unterschied der Verfahren darin, ob es eine zentrale Einheit gibt, die die Aufgaben verteilt, oder ob die Roboter untereinander solch eine Aufgabenverteilung vornehmen. Zentrale Lösungen (z.B. Shehory and Kraus [1998]) zielen dabei auf die Optimalität der Aufgabenausführung hin, wohingegen dezentrale Lösungen eine möglichst große Flexibilität des Mehrrobotersystems erreichen möchten. Dazu werden für die Task-Allocation sogenannte „Auktions-Modelle“ benutzt (siehe z.B. Koenig et al. [2008]). Dabei werden die verschiedenen Teilaufgaben innerhalb des Mehrrobotersystems versteigert. Meist werden als Währung die Kosten, die die Ausführung der Teilaufgabe mit sich bringt, benutzt. Der Roboter, für den die Teilaufgabe am wenigstens kostet, kann sie übernehmen. Ein Roboter, der z.B. aufgrund seiner Sensorausstattung eine Teilaufgabe nicht lösen kann, würde daher nicht für sie bieten, da die Kosten für ihn unendlich hoch wären. So wird eine schnelle Aufgabenausführung erreicht, ohne dass eine zentrale Stelle die Aufgaben vermitteln muss. Einen Überblick über verwendete Ansätze liefert auch die Arbeit von Parker [2008].

2.2.4 Exploration

Eine Gruppe von Robotern eignet sich sehr gut dafür, eine große unbekannte Umgebung schnell zu erkunden. Hier kann das Mehrrobotersystem seinen Vorteil, an mehreren Orten zugleich sein zu können, ausspielen. Um dies jedoch zu erreichen, müssen die Roboter verteilt werden. Burgard et al. [2000] definieren einen Informationsgewinn für jeden Punkt zwischen dem erkundeten und dem unbekannten Bereich (Frontiers). Ist schon ein Roboter auf dem Weg zu solch einem Punkt, wird der Informationsgewinn nahe Null liegen, ein weiterer Roboter muss sich nicht dorthin begeben. So werden die Roboter zwar nicht durch eine zentrale Einheit verteilt, da aber die Informationen (aktuelle Karte und Zielpunkte der einzelnen Roboter) zentral verwaltet werden, ist dies als zentral koordiniertes Planungsverfahren anzusehen. Diese sogenannten „Frontier-basierten“ Explorationsalgorithmen wurden mit verschieden Optimierungen versehen oder z.B. um Aspekte der Kommunikation erweitert (z.B. in Simmons et al. [2000] und Kulich et al. [2007]). Durch die Kombination von kooperativer Navigation und gemeinsamer Exploration kann die Genauigkeit der Karte erhöht werden (in Rekleitis et al. [1998]).

2.2.5 Coverage

Im Gegensatz zur Exploration geht es bei dem Coverage-Problem darum, eine möglichst große Fläche einer Umgebung (bekannt oder unbekannt) ständig mit den Sensoren der Roboter zu überdecken. In unbekannter Umgebung kann dies über lokale Entscheidungen gelöst werden. Batalin and Sukhatme [2002] schlagen dabei zwei Algorithmen vor, die die Roboter in der Umgebung verteilen. Dabei besteht die Koordination darin, dass die Roboter sich möglichst aus dem Weg gehen. Unter der Bedingung, dass die Umgebung unbekannt ist, die von den Robotern abgedeckte Fläche jedoch zusammenhängend ist, kann der Algorithmus aus Howard et al. [2002] genutzt werden. Dabei erzeugen die Roboter am Rande ihrer Sensorreichweite, wie bei einer Exploration, Grenzen. Auf diese Grenzen wird dann der nächste Roboter gesetzt. Da immer nur ein Roboter fährt, wird dabei eine wegzusammenhängende Fläche von den Robotern überwacht.

2.3 Koordination und Kooperation unter Nebenbedingungen

Die oben aufgeführten Aufgabenstellungen an ein Mehrrobotersystem zeigen verschiedene Probleme und beispielhaft vorgeschlagene Lösungen. Von besonderer Bedeutung in dieser Arbeit sind jedoch solche Aufgaben, bei denen zusätzlich zu Koordination und Kooperation noch weitere Einschränkungen beachtet werden müssen. Solche Einschränkungen können durch die Applikation gesetzt werden, aber auch im Ansatz der Lösung selbst begründet sein. So verlangt z.B. ein zentraler Ansatz, bei dem die Pläne online geändert werden, dass alle Roboter mit der zentralen Einheit in Funkverbindung stehen. Daher wäre es konsequent, in solchen Ansätzen dafür zu sorgen, dass die Aufrechterhaltung des Funkkontakts Teil der Koordinationsaufgabe ist. Eine solche Bewegungsplanung, die weitere, meist physikalische Anforderungen erfüllen muss, wird *Bewegungsplanung unter Nebenbedingung* genannt.
Wie die Systematiken oben schon zeigten, stehen bei der Suche von Lösungen für verschiedene Aufgaben unterschiedliche Eigenschaften bzw. Designs zur Verfügung, aus denen je nach Aufgabe ausgewählt werden kann. Im Folgenden werden die unterschiedlichen Eigenschaften näher betrachtet, verschiedene Arbeiten, die die jeweilige Eigenschaft implementieren, beispielhaft beschrieben und erläutert, warum bestimmte Designentscheidungen in der vorliegenden Arbeit so getroffen wurden. Die Eigenschaften, die hier vorgestellt werden, sind:

- Qualität der Nebenbedingung
- Komplexität der Koordination
- Zentrale oder dezentrale Planung
- Koordination als Graphenproblem

Zudem werden Arbeiten mit verschiedenen Anwendungsfällen beschrieben und gezeigt, wo Gemeinsamkeiten und Unterschiede zu der hier definierten Problemstellung liegen.

2.3.1 Qualität der Nebenbedingung

In der Koordination von Mehrrobotersystemen unter Nebenbedingungen ist die Art der Nebenbedingung von entscheidender Bedeutung. Unterschiede in den Arbeiten liegen dabei in der Definition der Nebenbedingung. Oft wird die Nebenbedingung als „Kommunikationsnebenbedingung“ bezeichnet, da sich die Vorteile dieser Nebenbedingung für Mehrrobotersysteme leicht erschließen.
So bezeichnen unter andern Spanos and Murray [2005] die betrachtete Nebenbedingung als Kommunikation. Dabei ist das Modell der Kommunikation ein einfacher Kreis mit einem bestimmten Radius um den Roboter. Alles innerhalb dieses Kreises kann mit der Funkverbindung erreicht werden, alles außerhalb nicht. Über die Definition eines „connectivity robustness“-Faktors kann eine Aussage über die Stabilität der Verbindungen innerhalb des Roboternetzwerkes gemacht werden.
Solch eine einfache Nebenbedingung wird häufig aus zwei Gründen genutzt. Zum einen werden die Versuche in einem Simulator und nicht auf realen Robotern durchgeführt. Dann stellt eine so vereinfachte Kommunikationsnebenbedingung in der Umsetzung kein Problem dar. Zum Zweiten werden solche Nebenbedingungen dann genutzt, wenn nicht der Einfluss der Nebenbedingung, sondern vor allem die Eigenschaften von veränderlichen Graphen betrachtet werden. So ergibt sich eine einfache Begründung, wie die Graphenstruktur innerhalb des Mehrrobotersystems entsteht, und die Auswirkungen von Bewegungen einzelner Knoten auf den Graphen können systematisch untersucht werden. Im Gegensatz zu Spanos und Murray wird in dieser Arbeit die Möglichkeit gegeben, auch komplexere Nebenbedingungen zu berücksichtigen. Dies wird zum einen durch einen Planer ermöglicht, der mit unterschiedlichen Nebenbedingungen ähnlich umgehen kann. Zudem war von vorneherein beabsichtigt, die Planungsergebnisse auf einem realen Robotersystem zu nutzen.
Arkin and Diaz [2002] beschreiben die eher selten verwendete „Sichtbarkeitsnebenbedingung“. In ihrer Arbeit soll eine Gruppe von Robotern eine Umgebung erkunden, dabei allerdings jederzeit Sichtkontakt untereinander halten. In einer Graphendarstellung bilden die Roboter dabei die Knoten, während es genau dann eine Kante zwischen zwei Robotern gibt, wenn sie sich gegenseitig sehen können. Dabei wird verlangt, dass der Graph eine einzige Zusammenhangkomponente bildet. Diese Nebenbedingung nennen Arkin und Diaz „operational constraint“. Ein Grund, warum die Sichtbarkeitsnebenbedingung nur selten verwendet wird, mag in der Tatsache liegen, dass diese Nebenbedingung binär ist. So kann die Sichtverbindung durch eine kleine Bewegung abreißen, was gerade in reaktiven Systemen, wie sie Arkin und Diaz hier beschreiben, nicht vorhergesehen werden kann. Daher implementieren sie eine „Rescue Funktion“, die die Roboter so zurück steuert, dass die Nebenbedingung wieder hergestellt ist. In der einfachen Va-

riante wird ein Roboter als Ankerpunkt bestimmt und bewegt sich nicht. Die anderen Roboter werden sequenziell bewegt, d.h. immer nur einer gleichzeitig. So kann, wenn die Nebenbedingung verletzt wird, einfach reagiert und der Roboter zurück bewegt werden. In der komplexeren Form des Algorithmus kann der Ankerroboter in bestimmten Grenzen bewegt werden. Dabei wird er von einem Quadranten zum nächsten bewegt. Diese Quadranten können jedoch nur definiert werden, wenn vorher schon gewisse Informationen über die Umgebung bekannt sind.
Drei verschiedenen Nebenbedingungen und deren Einflüsse auf die Exploration werden von Tuna et al. [2013] untersucht. Sie betrachten dabei den „Frontier"-basierten Ansatz, einen rollenbasierten Ansatz, wie in Pei et al. [2010], und einen „Auction"-basierten Ansatz, in dem die Roboter untereinander die nächsten Zielpunkte verhandeln.
Alle drei Arten von Nebenbedingungen fassen sie als Kommunikation zusammen:

- Eine einfache Distanznebenbedingung, in der Roboter dann miteinander kommunizieren können, wenn sie eine bestimmte Entfernung nicht überschreiten.
- Die Sichtbarkeitsnebenbedingung. Hier können die Roboter dann kommunizieren, wenn sie einander sehen können.
- Ein Kommunikationsmodell, das sowohl die Distanz, wie auch eventuell vorhandene Hindernisse berücksichtigt.

Die verschiedenen Kommunikationsmodelle werden in Tuna et al. [2013] mit den verschiedenen Explorationsverfahren verbunden und auf verschiedenen Computern mit unterschiedlicher Leistung getestet.
Es ist auffällig, dass sowohl die Arbeit von Tuna et al. wie auch die vorliegende Arbeit identische Nebenbedingungen betrachten. Beide Arbeiten entstanden parallel und ohne Kenntnis voneinander (beide Arbeiten sind in der gleichen Sonderausgabe des Ad-Hoc Journal erschienen Natalizio et al. [2013]). Dies ist ein Hinweis darauf, dass die in dieser Arbeit gewählten Beispiel-Nebenbedingungen sinnvoll sind. Dabei sind die Distanznebenbedingung und die Sichtbarkeitsnebenbedingung offensichtliche Nebenbedingungen, die Wahl des exakt gleichen Wellenausbreitungsmodells lässt sich vermutlich darauf zurückführen, dass in diesem Wellenausbreitungsmodell der Einfluss der Umgebung schnell berechnet werden kann und dennoch einen relativ guten Eindruck einer realen Funkverbindung vermittelt.
Es gibt allerdings auch Arbeiten, bei denen die Modellierung der Nebenbedingung im Vordergrund steht. So legt Mostofi [2008] in seiner Arbeit den Fokus auf eine realistische Modellierung des Kommunikationskanals, der als Nebenbedingung für die Navigation genutzt wird. Dabei wird insbesondere die mögliche starke Änderung der Signalstärke bei geringen Bewegungen betrachtet. Gerade in komplexen Umgebungen kann eine kleine Bewegung größere Änderungen in der Qualität des Kommunikationskanals bewirken. So können Mehrwegausbreitungen sich gegenseitig stören oder ein plötzlicher Abbruch dadurch verursacht werden, dass der Empfänger hinter einem Hindernis verschwindet. Mostofi vergleicht dann eine dezentralisierte koordinierte Navigation unter der Annahme

einer perfekten Kommunikation innerhalb eines bestimmten Radius (die Annahme, die in den meisten Arbeiten getroffen wird) mit einer Navigation bei der seinem realistischeren Modell genutzt wird. Dabei kann gezeigt werden, dass die koordinierte Navigation davon profitieren kann, wenn sie z.B. Punkte in der Umgebung ausnutzt, an denen die Kommunikation besser ist, als sie nach einem einfachen Modell sein würde.
Insgesamt zeigt sich ein großes Spektrum an möglichen Nebenbedingungen. Zudem zeigt sich, dass die Nebenbedingung meist das Koordinationsverfahren beeinflusst, insbesondere dann, wenn die Roboter reaktiv handeln. Daher wurde in dieser Arbeit bewusst darauf aufgebaut, eine generische Sicht auf die Nebenbedingungen zu behalten. Durch die gewählte Graphendarstellung kann eine große Gruppe von Nebenbedingungen, in dieser Arbeit als *spatiale Nebenbedingungen* bezeichnet, betrachtet werden. Von diesen sind die Beispielnebenbedingungen Distanz, Sichtbarkeit und Wellenausbreitungsmodell (Kommunikation) nur die offensichtlichsten.

2.3.2 Komplexität der Koordination

In den unterschiedlichen Arbeiten über Bewegungsplanung für Mehrrobotersysteme unter Nebenbedingung fallen auch die unterschiedlichen Grade der Koordination auf. Ein Beispiel, bei dem nur ein einziger Parameter koordiniert werden muss, ist das oben genannte Pfadkoordinierungs-Problem, wie z.B. in Abichandani et al. [2009] besprochen. Dabei wird die Koordination nur über den Geschwindigkeitswert der Roboter erreicht. Pfad und Zielpunkte sind vorgegeben. Dennoch kann darüber nicht nur die Ausführungszeit optimiert werden, auch eine Kommunikationsbedingung kann eingehalten werden.
Mehrere zu koordinierende Parameter sind in Notarstefano et al. [2006] zu finden. Hier können die Bewegungen, d.h. die Pfade der Roboter, auch verändert werden. Dazu wird die Dynamik der Roboter als Differentialgleichung zweiter Ordnung aufgefasst. Die Nebenbedingung wird hier als R-disk-Graph (Distanznebenbedingung) modelliert.
Einen allgemeinen Ansatz, eine Kommunikationsnebenbedingung in verschiedenen Koordinationsverfahren einzubringen, verfolgen Van Tuan et al. [2009]. Dabei wird angenommen, dass die Roboter untereinander in einem MANET (Mobile Ad Hoc Network) kommunizieren. Solch ein Netz liefert bestimmte Informationen über das Routing innerhalb des Netzes, d.h. die Pfade, die eine Nachricht vom Sender zum Empfänger nimmt. Diese Information nutzen Van Tuan et al. aus, um den Zustand des Netzes abzuschätzen und für die Navigation zu verwenden. So führt jeder Roboter eine Liste von Pfaden zu verschiedenen Robotern des Mehrrobotersystems. Durch Veränderungen der Umgebung oder durch Bewegungen des Roboters kann sich die Topologie des Netzwerkes ändern. Solche Änderungen resultieren in Updates der Listen. Anhand der Liste können verschiedene Eigenschaften des aktuellen Kommunikationsnetzes von jedem Roboter dezentral gewonnen werden. Zunächst kann sehr leicht nachverfolgt werden, welche Verbindung durch Bewegung unterbrochen werden darf, ohne dass das Netz in

mehrere Teilkomponenten zerfällt. Aber auch Eigenschaften, die die Ausfallsicherheit erhöhen, können überprüft werden. So kann die Bi-konnektivität geprüft werden, d.h. ob ein Roboter ausfallen kann, ohne dass das Netz zerfällt. Auch die Überprüfung, ob das Netzwerk doppelt verbunden ist, kann mit wenig Aufwand dezentral vorgenommen werden. Dabei wird festgestellt, ob in dem Netz eine beliebige Kante gelöscht werden kann, ohne dass das Netz zerfällt. Solche Informationen können dann genutzt werden, um die Navigation zu verbessern. In Van Tuan et al. [2009] wird diese Netzwerküberwachung genutzt, um das Navigationsverfahren aus Rooker and Birk [2007] so zu modifizieren, dass es keine zentrale Koordinationseinheit mehr benötigt.
Innerhalb von Mehrrobotersystemen können die unterschiedlichsten Parameter genutzt werden, um eine Koordination zu erreichen. In der hier vorliegenden Arbeit sind nur einige Zielpunkte und die Nebenbedingung vorgegeben. Mit diesen Angaben kann der Planer einen Plan erzeugen, der einen Roboter auf jeden der Zielpunkte bringt, ohne die Nebenbedingung zu verletzen. Dazu werden die Pfade berechnet und Zwischenziele bestimmt, auf denen Roboter stehen bleiben müssen. Durch die große Gruppe von Nebenbedingungen und der Möglichkeit, Höhenkarten zu verwenden, kann der Planer damit in vielen Situationen, in denen Roboter auf einer Karte gezielt positioniert werden müssen, verwendet werden.

2.3.3 Zentrale oder dezentrale Planung

In der Literatur finden sich Ansätze von Koordination, bei der der Koordinationsmechanismus unterschiedlich ausgeführt sein kann. So kann die Koordination zentral oder dezentral durchgeführt werden. Ein Beispiel für eine zentrale Koordination findet sich in Rooker and Birk [2007]. Hier wird eine Methode zur Exploration unter der Distanznebenbedingungen beschrieben. Einer zentralen Instanz liegen alle Informationen der Roboter (d.h. ihre Position und die schon erkundete Karte) vor. Diese Instanz entscheidet, welche mögliche Aktion des Mehrrobotersystems als nächstes ausgeführt werden soll. Dazu werden die Bewegungsbefehle auf acht Himmelsrichtungen und einen „stehenbleiben“-Befehl diskretisiert. Nun kann ein endlicher Satz an möglichen Bewegungsbefehlen für das ganze MRS definiert werden. Da dieser Satz zu viele Befehle enthält und nicht in jedem Zeitschritt vollständig bewertet werden kann, wird eine zufällige Teilmenge ausgewählt, die mit Hilfe einer Fitnessfunktion bewertet wird. Der am besten bewertete Befehl wird an die Roboter verteilt, die diesen dann ausführen. In den Simulationsexperimenten zeigte sich, dass in den typischen Innenraumumgebungen Deadlocks auftreten können, aus denen sich das System nicht befreien kann. Die zentrale Koordinationseinheit plant immer nur einen Schritt voraus, was zur beschriebenen Deadlock-Situation führt. Mit Hilfe einer expliziten Deadlockerkennung konnte dieses Problem behoben werden. Dieses Vorgehen hat den Vorteil, dass nach einer Änderung der Umgebung nicht ein vorher errechneter Plan verworfen werden muss. Dennoch wird in dieser Arbeit ein zentraler Planer gewählt, der einen Plan für die gesamte Aktionszeit des Mehr-

robotersystems erstellt. Auf diese Art, können Deadlocks, die durch die Nebenbedingung entstehen können, von vornherein ausgeschlossen werden.
Einen dezentralen Ansatz beschreiben Abichandani et al. [2009]. Hierbei handelt es sich um einen verhaltensbasierten Ansatz. Die Roboter sollen dabei eine Umgebung erkunden, während sie die Sichtbarkeitsnebenbedingung erfüllen. Ein Roboter erfüllt die Rolle eines Ankerpunktes, der sich je nach Algorithmus nicht oder nur in bekannten Bahnen bewegt. Andere Roboter erfüllen die Rolle der Erkunder. Hier gibt es sowohl aktive wie passive Erkunder. In der einfachsten Version darf immer nur ein Roboter aktiv fahren, in den komplexeren Algorithmen, die jedoch ein bestimmtes Weltwissen verlangen, auch mehrere. So können die Roboter mit relativ wenig Kommunikation die Umgebung erkunden, ohne dass eine zentrale Einheit Aufgaben verteilen muss.
Es gibt durchaus Unterschiede in der Mächtigkeit der Ansätze zentral oder dezentral. Da die zentralen Ansätze normalerweise vollständig sind, d.h., wenn es eine Lösung, gibt finden sie auch eine, besteht die Herausforderung darin, einen Plan zu finden, der möglichst nahe am optimalen Plan ist. Der Schwerpunkt der Arbeiten um die dezentralen Koordinierungsverfahren unter Nebenbedingungen liegt meist darauf, überhaupt eine Lösung zu finden. So soll ohne die zentrale Instanz das Verhalten der Roboter robust so konvergieren, dass die Aufgabenstellung gelöst wird. Spanos and Murray [2005] untersuchen, welche Bewegungen mit dem dezentralen Mehrrobotersystem bei Beibehaltung einer Nebenbedingung möglich sind, ohne die Stabilität des Netzwerkes aus Robotern zu gefährden. Grundsätzlich ist dabei die Berechnung der Bewegung, die notwendig ist, die Topologie des Roboternetzwerkes von einer Konfiguration in eine andere zu bringen, dezentral angelegt. Spanos und Murray zeigen jedoch, dass diese dezentrale Lösung ihre Grenzen hat, da bestimmte Topologieübergänge einen globalen Planer benötigen.

2.3.4 Koordination als Graphenproblem

Durch die Nebenbedingung entsteht eine bestimmte Beziehung zwischen den Robotern. Meist wird dabei betrachtet, zwischen welchen Robotern eine Nebenbedingung erfüllt ist und zwischen welchen nicht. Daraus ergibt sich natürlicherweise ein Graph, der, wenn er aus einer einzigen Zusammenhangskomponente besteht, anzeigt, dass die Nebenbedingung vom ganzen Mehrrobotersystem eingehalten wird. Dabei können Eigenschaften, die von allgemeinen Graphen bekannt sind, auf die Koordination des Mehrrobotersystems übertragen werden.
So bilden die Roboter in De Gennaro and Jadbabaie [2006] einen Graph, in dem die Roboter die Knoten darstellen und Kanten dann existieren, wenn die Roboter nahe genug beieinander stehen. Solch ein Graph wird hier „R-disk graph“ genannt. Durch einen kontrolltheoretischen Ansatz, bei dem die „algebraic connectivity“ (Details in Godsil and Royle [2001]) maximiert wird, ist der gebildete Graph stabil. Damit sind die Roboter so miteinander vernetzt, dass der Ausfall eines Roboter die Kommunikation im gesamten MRS nicht beeinträchtigt.

Basu and Redi [2004] betrachten ein Netzwerk von vielen Robotern. Sie gehen davon aus, dass diese Roboter als Ad-Hoc-Netzwerk kommunizieren. Das bedeutet, die Topologie des Netzwerkes ändert sich, je nachdem wie die Roboter sich bewegen. Jeder Roboter ist dann in der Lage mit jedem Roboter innerhalb seiner Reichweite zu kommunizieren. Um mit einem weiter entfernten Roboter zu kommunizieren, müssen andere Roboter als Zwischenstation genutzt werden. So kann die Kommunikation zwischen den Robotern wiederum als Graph aufgefasst werden, in dem jeder Roboter mit jedem Roboter kommunizieren kann, so lange im Graph ein Weg zwischen den beiden existiert. Je nach Position der Roboter entstehen dabei jedoch Knoten im Netz, die, wenn sie ausfallen, den Kommunikationsgraphen in zwei oder mehr Teile zerfallen lassen. Solche „Single Points of Failure“ gilt es bei Basu und Redi zu verhindern. Durch die koordinierte Bewegung der Roboter sollen solche kritischen Punkte im Graphen aufgelöst werden. Dazu werden zwei Algorithmen vorgeschlagen: zum einen der „Kontraktionsalgorithmus“ und zum anderen eine „Blockbewegung“. Beim Kontraktionsalgorithmus werden alle Roboter so lange auf den virtuellen Schwerpunkt zu bewegt, bis alle Roboter so nahe beieinander stehen, dass kein „Single Point of Failure“ mehr existiert. Dieser sehr einfache Algorithmus hat den Nachteil, dass sich die Topologie des gesamten Kommunikationsnetzes während der Kontraktion stark ändern kann. Der zweite Algorithmus versucht durch eine Blockbewegung, die Topologie nicht zu drastisch zu verändern. Dabei werden die Teile des Netzwerkes, die durch den Ausfall eines Roboters abgetrennt werden können, als gesamter Block bewegt. Dieser Block bewegt sich auf den nächsten Roboter, der nicht mit dem Block verbunden ist, zu. Wird der Block mit diesem Roboter verbunden, ohne dabei die andere Verbindung zu verlieren, ist der „Single Point of Failure“ behoben. Es kann jedoch auch passieren, dass eine Art Pendelbewegung entsteht, nämlich dann, wenn die Verbindung zwischen Block und Rest des Netzwerkes abreißt, bevor der Block die Verbindung zu dem anderen Roboter herstellt. Solche Pendelbewegungen werden abgefangen, indem dann der Block auf den nächsten Roboter zufährt, der mit dem nächsten unverbundenen Roboter verbunden ist.

Auf bekannte Eigenschaften in Graphen greift auch das in dieser Arbeit vorgestellte Verfahren zurück. Wie in der Arbeit von Pei et al. [2010] schon angedeutet, besteht ein direkter Zusammenhang zwischen den optimalen Positionen der Roboter und Steinerbäumen. Während jedoch in Pei et al. [2010] eine spezielle Heuristik, die „Steinerized Minimum Spanning Tree“ genutzt wird, die der Tatsache geschuldet ist, dass die Roboter zugleich explorieren, kann in dieser Arbeit die Äquivalenz zwischen Steinerbäumen und optimalen Positionierungen bewiesen werden (siehe Lemma 4.13 in Kapitel 4.2.1). So konnte ein erster Algorithmus zur Planung, der STPlaner, entwickelt und als Messlatte für weitere Entwicklungen verwendet werden.

2.3.5 Koordination in Mehrrobotersystemen und ihre Anwendungen

Wie bei der Koordination von MRS werden auch bei der Koordination unter Nebenbedingungen verschiedene Aufgaben an das Mehrrobotersystem gestellt. Dabei ist die Exploration die bei weitem häufigste Anwendung. Hier sollen die Roboter koordiniert eine Umgebung erkunden, ohne dabei die Nebenbedingung zu verletzen.

So wird von Doniec et al. [2009] eine formalisierte Lösung für die Exploration unter der Kommunikationsnebenbedingung vorgeschlagen. Dabei wird das Problem als „constraint satisfaction problem“ (CSP) dargestellt. Um eine dezentrale Lösung zu erreichen, wird es auf das sogenannte „distributed constraint satisfaction problem“ (disCSP) erweitert. Um das disCSP zu formulieren, müssen zunächst die Nebenbedingungen definiert werden. Zusätzlich zur Kommunikationsnebenbedingung, die fordert, dass zu jedem Zeitpunkt jeder Roboter mit jedem anderen Roboter kommunizieren kann, fügen Doniec et al. zwei weitere hinzu. Zum einen dürfen sich die Sensorbereiche zweier Roboter nicht überlappen. Mit dieser Nebenbedingung wird eine Verteilung der Roboter im Raum erreicht. Zum zweiten wird verlangt, dass jeder Roboter mit jedem Schritt unbekannte Umgebung erkundet. Damit wird die Erkundung vorangetrieben. Um das disCSP zu lösen, müssen die Aktionen der Roboter diskretisiert werden, da der verwendete Algorithmus nicht auf einem kontinuierlichen Suchraum arbeiten kann. Die Bewegungen pro Zeitschritt sind auf 8 Himmelsrichtungen festgelegt. In der Simulation wird gezeigt, dass die Robotergruppe auf diese Weise in der Lage ist, Umgebungen zu explorieren.

Pei et al. [2010] erweitern ihre Exploration ebenfalls um eine Kommunikationsnebenbedingung. Sie beachten dabei nicht nur, dass die Roboter untereinander kommunizieren können, sondern auch, dass die zur Verfügung stehende Bandbreite ausreicht, um die gewonnenen Daten aus der Exploration an eine Basisstation zu schicken. Diese Basisstation bildet auch die zentrale Einheit für das zentral angelegte Koordinationsverfahren. Für die Exploration selber wird ein Frontier-basierter Ansatz genutzt. Besonders ist hier, dass nicht alle Roboter für die Exploration zuständig sind. Je nachdem, wie weit die Grenzen zur unbekannten Umgebung von der Basisstation entfernt liegen, wird eine Anzahl Roboter als Relais-Roboter genutzt. Diese Roboter überbrücken die Entfernung zwischen Erkundungsrobotern und Basisstation und bieten damit die Möglichkeit, die Gesamtreichweite des Mehrrobotersystems zu erweitern. So entsteht hier ebenso wie in der vorliegenden Arbeit das Problem, dass eine Verbindung zwischen dem Robotersystem und einer Basisstation geschaffen werden muss. Dazu werden Relaisroboter eingesetzt, die entsprechend positioniert werden müssen. Da in der vorligenden Arbeit die Zielpunkte einiger Roboter bekannt sind, kann der Zusammenhang zwischen optimaler Positionierung und Steinerbäumen direkt ausgenutzt werden. Zudem wird bei Pei et al. in der Koordination der Roboter eine feste Zuordnung von Roboter in Explorationsroboter und Relaisroboter benutzt, die die Reichweite des MRS beschränkt. So können Explorationsroboter nicht als Relaisstationen genutzt werden, was die Flexibi-

lität einschränkt. In der vorliegenden Arbeit können Roboter auf von Benutzer gesetzten Zielpunkten zugleich als Relaisstationen für weitere Roboter agieren, was die Anzahl benötigter Roboter reduziert und die Reichweite des MRS erhöht.

Auch die koordinierte Navigation unter Nebenbedingungen, bei dem mehrere Roboter von verschiedenen Startpunkten zu Zielpunkten gelangen sollen, ohne die Nebenbedingung zu verletzten, wird betrachtet. In Abichandani et al. [2009] wird dies als „Pfadkoordinierungsproblem mit der Nebenbedingung Kommunikation“ bezeichnet. Dabei sind sowohl Start- und Zielpunkt wie auch der Pfad dazwischen für jeden Roboter gegeben. Die Koordination beschränkt sich auf die Bestimmung der Geschwindigkeit jedes Roboters zu jedem Zeitpunkt. Da die Pfade sich üblicherweise kreuzen, können durch das Setzten der richtigen Geschwindigkeiten Kollisionen vermieden werden. Ziel ist es, die Geschwindigkeiten der Roboter so zu bestimmen, dass das gesamte System so schnell wie möglich an die Zielorte gelangt. Abichandani et al. stellen zudem eine Kommunikationsnebenbedingung vor, die vom gesamten System eingehalten werden muss, d.h. die Roboter müssen so positioniert sein, dass über eine Ad-hoc Struktur jeder Roboter mit jedem Roboter kommunizieren kann. Das Problem wird als „discrete time nonlinear programming problem (NLP)“ formuliert und gelöst. Dabei führen Abichandani et al. einen Mechanismus ein, der die Kommunikation nur dann berücksichtigt, wenn dies auch nötig ist, und somit bei der Berechnung Zeit spart. Die Aufgabe Roboter so zu koordinieren, dass sie zu ihren Zielpunkten gelangen, ohne die Nebenbedingung zu verletzten, zeigt einige Parallelen zur in dieser Arbeit vorgestellten Aufgabe. Auch hier sind vom Benutzer Ziele und der Startpunkt (an der Basisstation) vorgegeben. Zunächst wird auch davon ausgegangen, dass die Umgebung bekannt ist, so dass die Pfade für die Roboter zu den Zielen im Prinzip leicht zu berechnen sind. Hier zeigt sich jedoch der Unterschied: Die hier neu definierte Aufgabe verkompliziert sich dadurch, dass zusätzliche Zielpunkte für Roboter berechnet werden müssen. Diese sind die Zielpunkte für die Relaisroboter, die nötig sind, um andere Roboter an die vorgegebenen Zielpunkte zu bringen, ohne die Nebenbedingung zu verletzen. Diese Relaisstationen müssen von vornherein beachtet werden, um einen möglichst guten Weg für alle Roboter zu finden, bei dem möglichst wenige Roboter benutzt werden müssen.

Eine weitere Anwendung ist die des „multi-robot routing“. Dabei sollen die Roboter des MRS verschiedene Ziele besuchen. Üblicherweise gibt es dabei mehr Ziele als Roboter, so dass durch eine geschickte Zuordnung der Roboter zu den Zielen die Ausführungsgeschwindigkeit oder die zu fahrende Strecke minimiert wird. In den Arbeiten Arrichiello et al. [2010] und Jnaneshwar et al. [2009] werden Nebenbedingungen für die Navigation von zwei Booten (USV, umanned surface vehicle) beschrieben. Auch hier ist die Nebenbedingung als Kommunikationsnebenbedingung bezeichnet, wird jedoch als Distanznebenbedingung modelliert. Das führt dazu, dass die Roboterboote nicht weiter als eine bestimmte Distanz voneinander entfernt sein dürfen. Gelöst wird das Problem in einem Schichtenmodell, in dem jedes der Boote eine obere „Aufseher-Schicht“ besitzt, die verhindert, dass darunter liegende verhaltensbasierte Ansätze das Boot außerhalb der

erlaubten Distanz bewegen. In Experimenten mit realen Systemen wurde gezeigt, dass dieser Ansatz eine Zeitersparnis bei der Ausführung der Aufgabe gegenüber der Aufgabenerfüllung mit nur einem Roboter von etwa 30% erreicht.
Auch in den Arbeiten von Mosteo et al. [2008, 2009] wird das sogenannte „multi-robot routing"-Problem behandelt. Dabei ist eine Menge von Zielpunkten gegeben, wobei jeder Zielpunkt von einem Roboter genau einmal angefahren werden soll. Gesucht ist der beste, d.h. hier der kürzeste Weg. Damit muss jedem Roboter eine Menge von Zielpunkten gegeben werden, sowie eine Reihenfolge, wie er sie abfahren soll. Mosteo et al. erweiterten dieses Problem um die Einhaltung der Kommunikation zu einer unbeweglichen Basisstation. Es wird angenommen, dass ab einer bestimmten Signalstärke Kommunikation zwischen zwei Robotern möglich ist. Um diese Kommunikation aufrecht zu erhalten, wird ein reaktives System vorgeschlagen, das auf sogenannten „Spring-Dampern" beruht. Dabei handelt es sich um virtuelle Verbindungen zwischen zwei Robotern, die die Roboter zueinander ziehen, wenn ein Messwert zwischen den Robotern einen bestimmten Bereich verlässt. Um nun einen Zielpunkt zu erreichen, der von der Basisstation weit entfernt ist, benötigt es eine Gruppe von Robotern. In einer Art „Leader-Follower"-System fährt ein Roboter der Gruppe diesen Zielpunkt an, während er durch die virtuellen Verbindungen weitere Roboter hinter sich herzieht. So entsteht eine Kette von Robotern, die eine Verbindung zur Basisstation ermöglicht. Da im Voraus jedoch nicht bekannt ist, wie viele Roboter in solch einer Kette benötigt werden, um den jeweiligen Zielpunkt zu erreichen, kann dies nur geschätzt werden. Wird die Anzahl benötigter Roboter zu hoch eingeschätzt, wird die Ausführungszeit unnötig vergrößert. Wird die Zahl jedoch zu klein eingeschätzt, kann der Zielpunkt nicht erreicht werden und muss später mit einer größeren Gruppe nochmal angefahren werden. Damit erhöht sich auch in diesem Fall die Ausführungszeit. Gerade bei dieser Variante, in der eine bestimmte Anzahl Roboter zur Verfügung stehen, sind Parallelen mit dem Anwendungsfall dieser Arbeit zu sehen. Auch hier müssen die Roboter bestimmte vorgegebene Zielpunkte erreichen und andere Roboter als Relaisroboter dazu nutzen die Kommunikation aufrecht zu erhalten. Während aber im „multi-robot routing Problem" die Zielpunkte nacheinander angefahren werden, da es mehr Zielpunkte als Roboter gibt, muss in der vorliegenden Arbeit das MRS die gegebenen Punkte gleichzeitig besetzten, was die Pfadplanung gegenüber der Aufgabenverteilung betont.

3 Grundlagen

3.1 Graphentheroetische Grundlagen

Der in dieser Arbeit vorgeschlagene Ansatz zur Lösung des Problems der koordinierten Navigation unter spatialen Nebenbedingungen nutzt einige bekannte Algorithmen aus der Graphentheorie. Da diese teilweise essentiell für das Verstehen der genutzten Algorithmen sind, sollen sie hier kurz vorgestellt werden.

3.1.1 Nomenklatur

Bei der Nomenklatur wurde sich vornehmlich an die Nomenklatur aus Latombe [1991] und Klein [2005] gehalten:

1. Ein Roboter wird mit $\mathcal{A}$ bezeichnet. Sind mehrere Roboter angesprochen, so werden sie mit $\mathcal{A}_i$ ($i = 1, 2, \ldots$) referenziert.
2. Ein Pfad wird mit Π bezeichnet. Dabei besteht ein Pfad üblicherweise aus einer Reihe von Knoten.
3. Die Distanz zwischen zwei Punkten x und y im Raum ist mit $||x-y||$ beschrieben.
4. Ein *Graph* wird mit G bezeichnet. Er besteht aus der Knotenmenge V, wobei ein einzelner Knoten mit v bezeichnet wird, und einem System von Kanten E mit einer einzelnen Kante e.
5. Positionen in der realen Umwelt werden mit $\mathbf{a}, \mathbf{b}, \mathbf{c}, \ldots$ bezeichnet.

3.1.2 Das Steinerbaum-Problem

Das Steinerbaum-Problem ist ein Problem aus der Graphentheorie, das vor allem in der technischen Informatik (Layout von Platinen), aber auch in der Netzwerktechnik eine Rolle spielt. Es wird wie folgt definiert: Gegeben sei ein Graph $G = (V, E)$ und eine Menge von Knoten $S \subseteq V$ (Terminale). Dann ist der Steinerbaum der Graph, der alle Knoten aus S verbindet und dabei ein minimales Kantengewicht hat. Um alle Knoten auf S miteinander zu verbinden, werden im Allgemeinen mehr Knoten benötigt, als S enthält. Diese zusätzlichen Knoten werden Steinerknoten genannt. Das Steinerbaum-Problem ist NP-vollständig. Daher werden Heuristiken benötigt, um große Instanzen in akzeptabler Zeit berechnen zu können.

Steinerbaum-Heuristik

Eine der bekanntesten Heuristiken zur Bestimmung eines Steinerbaums ist der Algorithmus von Mehlhorn. Der berechnete Steinerbaum ist maximal zweimal so schlecht wie der optimale. Dabei hat der Algorithmus eine Laufzeit von $O(|E| + |V|log|V|)$, wobei E die Menge aller Kanten und V die Menge aller Knoten im Graph ist.
Der Algorithmus von Mehlhorn besteht im Grunde aus 5 Schritten, wobei die Schritte 2 bis 5 aus Kou et al. [1981] entnommen sind. Die Leistung Mehlhorns ist der Beweis, dass in Schritt 1 ein vollständiger Distanzgraph nicht notwendig ist und somit dort ein großer Teil des Rechenaufwandes gespart werden kann. Nach Mehlhorn [1988] kann der Steinerbaum auf folgende Weise berechnet werden:

1. Sei:
   - $S \subseteq V$; die sogenannten Terminale
   - $d_k(u, v)$ ist die Distanz zwischen den Knoten u und v im Graphen G_k
   - Für jedes $s \in S$ ist $N(s)$ die Menge von Knoten in V, die näher an s sind als an alle anderen Knoten in S
   - $N(s) \cap N(t)$ mit $s, t \in S; s \neq t$ ist leer,
     d.h. $v \in N(s) \Rightarrow d_1(v, s) \leq d_1(v, t)$ für alle $s, t \in S$
2. Konstruiere den Graphen $G'_1 = (S, E'_1, d'_1)$.
   Dabei ist $E'_1 = \{(s, t); s, t \in S$, und eine Kante $(u, v) \in E$ wenn $u \in N(s), v \in N(t)\}$.
   $d'_1(s, t) = min\{d_1(s, u) + d_1(u, v) + d_1(v, t); (u, v) \in E, u \in N(s), v \in N(t)\}$.
3. Finde $G_2 = MST(G'_1)$ (Minimal spanning tree).
4. Baue den Teilgraphen G_3 auf, indem alle Kanten in G_2 durch die Pfade aus G ersetzt werden, die sie repräsentieren.
5. Finde $G_4 = MST(G_3)$.
6. Baue den Steinerbaum G_5 aus G_4 auf, indem so lange alle Blätter aus G_4 gelöscht werden, bis alle Blätter Terminale sind.

Der Mehlhorn-Algorithmus löst das Steinerbaum-Problem auf Graphen. In Hwang and Richards [1992] werden weitere Heuristiken für andere Instanzen des Steinerbaum-Problems vorgestellt. Hwang und Richards benennen vier Ausprägungen des Steinerbaum-Problems:

- Das euklidische Steinerbaum-Problem: Hierbei sind die Terminale, d.h. die zu verbindenden Punkte, in der euklidischen Ebene eingebettet. Gesucht wird nun ein Baum, der diese Punkte verbindet. Dabei können weitere Knoten (Steinerknoten) in der Ebene erzeugt werden.
- Das Steinerbaum-Problem auf Graphen: Hierbei ist der Graph, auf dem der Steinerbaum berechnet werden soll, vorgegeben. Steinerknoten können nur aus dem Satz Knoten, den der Graph bereitstellt, gewählt werden.
- Bei dem rectilinearen Steinerbaum-Problem handelt es sich um das Steinerbaum-Problem in der Ebene mit der Manhattan-Metrik.
- Phylogenetic Trees: Hierbei wird das Steinerbaum-Problem auf phylogenetischen Bäumen angewandt. Diese Bäume sind eine Darstellungsform der Entwicklung der Art im evolutionären Prozess. Steinerbäume können hier genutzt werden, um Sequenzanalysen durchzuführen.

Dynamische Steinerbäume

Das oben beschriebene Problem, Steinerbäume zu finden, kann auch auf dynamische Graphen erweitert werden. Unter dynamischen Graphen versteht man dabei Graphen, die sich im Laufe der Zeit ändern, also z.B. Kanten oder Knoten verlieren. Das Problem der dynamischen Steinerbäume bezieht sich darauf, dass zunächst ein Steinerbaum für einen Graph und einen Satz von Terminalen berechnet wird. Danach ändert sich der Graph. Nun muss der ehemalige Steinerbaum dieser Gegebenheit angepasst werden, möglichst ohne dass er vollständig neu berechnet wird.
In Bauer and Varma [1995] wird das Problem der dynamischen Anpassung von Steinerbäumen an der realen Applikation des „multicast routings“ beschrieben. Hierbei existiert ein Funknetzwerk mit vielen verschiedenen Teilnehmern (Knoten) und verschiedenen Verbindungen untereinander (Kanten). Dabei können jedoch nicht alle mit allen direkt kommunizieren, sondern müssen andere Teilnehmer als Relaisstationen nutzen. Bei der Multicast-Übertragung soll nun von einem Sender an mehrere Empfänger ein Datenpaket gesendet werden. Daher ist es hier notwendig, einen Steinerbaum mit Sender und Empfängern als Terminale zu berechnen. Nun ist es möglich, dass sich das Funknetz über die Zeit verändert. Hier betrachten Bauer und Varma den Fall, dass neue Teilnehmer hinzukommen können oder alte Teilnehmer aus dem Funknetz ausscheiden. Dafür wird der Algorithmus ARIES vorgeschlagen und Referenzalgorithmen gegenüber gestellt. ARIES beruht bei der Berechnung der dynamischen Steinerbäume darauf, dass die „Beschädigung“ beim Hinzufügen oder Löschen von Knoten beobachtet wird. D.h. es wird abgeschätzt, wie viel schlechter der Steinerbaum durch die Änderung des Netz-

werkes geworden ist. Wird ein bestimmter Schwellwert überschritten, wird der aktuelle Steinerbaum angepasst.
In Escoffier et al. [2009] wird davon ausgegangen, dass anfänglich ein optimaler Steinerbaum zur Verfügung steht. Der vorgestellte Algorithmus sorgt dann dafür, dass beim Hinzufügen oder Löschen von Knoten eine gute Lösung des Steinerbaums beibehalten wird. Dabei werden verschiedene Fälle untersucht: so können ein oder mehrere Knoten hinzugefügt werden, wobei diese Knoten Terminale sein können oder nicht. Für den Fall des Löschens von Knoten ist dann gezeigt worden, dass es Fälle gibt, in denen der Steinerbaum komplett neu berechnet werden muss.
Nicht nur auf veränderlichen Graphen, sondern auch auf Umgebungen im euklidischen Raum können dynamische Steinerbäume berechnet werden. In Alon and Azar [1992] wird dabei angenommen, dass die Terminale nicht von Anfang an bekannt sind, sondern nacheinander präsentiert werden. So muss für jeden neu präsentierten Punkt der Steinerbaum angepasst werden. Dabei werden allerdings nur weitere Kanten in den Steinerbaum hinzugefügt. Es kann sich die Topologie des Steinerbaums also nicht grundsätzlich durch das Hinzufügen eines Terminals ändern. Diese Problemstellung wird das „on-line planar Steiner-tree problem“ genannt. Alon und Azar zeigen in dieser Arbeit obere und untere Schranken für die Güte einiger bekannter Algorithmen.

3.2 Grundlagen aus der Kommunikation

Wie in den oben zitierten Arbeiten gezeigt, ist die Kommunikationsnebenbedingung die wohl wichtigste Nebenbedingung für ein Mehrrobotersystem. Daher wird die vorliegende Arbeit einen Schwerpunkt auf diese Nebenbedingung legen. Wie in Tuna et al. [2013] kann jede in dieser Arbeit genutzte Nebenbedingung als Modell einer Kommunikation angesehen werden. Explizit als Kommunikationsnebenbedingung wird hier jedoch ein einfaches Wellenausbreitungsmodell bezeichnet. Dieses Modell wird in Abschnitt 3.2.1 beschrieben.
Neben der expliziten Nutzung des Wellenausbreitungsmodells für die Schätzung der Kommunikation wird ein weiteres Konstrukt aus der Kommunikationstechnik genutzt, die „Multipoint-Relays“. Sie werden jedoch in einen neuen Zusammenhang gebracht und im lokalen Suchalgorithmus, welcher in Kapitel 4.2.2 entwickelt wird, zur schnellen Durchmusterung eines Graphen genutzt. Das Konzept der Multipoint Relais (MPR) und ein Algorithmus zur Bestimmung einer MPR Gruppe wird in Abschnitt 3.2.2 beschrieben.

3.2.1 Wellenausbreitungsmodell

Die Vorhersage, wie sich die Funkverbindung zwischen zwei Systemen, wie z.B. Robotern, entwickelt und ob Kommunikation möglich ist, verbessert die Ergebnisse in der

Koordination. Allerdings ist die Physik der Wellenausbreitung sehr komplex und gerade in kleinen Bereichen („small scale“) verhält sich der reale Funk nahezu chaotisch. So können schon kleine Bewegungen für eine starke Änderung der Signalstärke sorgen. Wellenausbreitungsmodelle, die solche Effekte berücksichtigen können, benötigen zum einen ein sehr genaues Modell der Umgebung und zum anderen viel Rechenzeit. Allerdings können auch einfachere Modelle genutzt werden, wenn die Tendenz bzw. ein grobes Ergebnis ausreicht oder nicht genug Rechenzeit und -kapazität vorhanden ist. Bekannte Modelle sind das „Rayleight fading modell“ [Hashemi, 1993], das „Rician distribution modell“ [Rice, 1944] und das „Floor Attenuation Factor Modell“ [Seidel and Rappaport, 1992]. Auf letzterem basiert das Modell von Bahl und Padmanabhan [Bahl and Padmanabhan, 2000], welches für eine Indoorlokalisierung eines Senders innerhalb eines Gebäudes genutzt wurde. Ein sehr ähnliches Modell wird in dieser Arbeit verwendet, um die Signalstärke zwischen zwei Punkten auf einer Oberfläche zu schätzen. Das Modell von Bahl und Padmanabhan lautet:

$$P(d)[dBm] = p(d_0)[dBm] - 10nlog\frac{d}{d_0} - \begin{cases} nW * \mathit{WAF}, & \text{falls } nW < C \\ C * \mathit{WAF}, & \text{sonst} \end{cases}$$

mit:

n	:=	„path loss“ Faktor
p(d_0)	:=	Signalstärke in Referenzdistanz d_0
d	:=	Distanz zwischen Sender und Empfänger
C	:=	Anzahl von Hindernissen (z.B. Wände), die maximal betrachtet werden
nW	:=	Anzahl der Hindernisse auf dem Weg zwischen Sender und Empfänger
WAF	:=	Dämpfung am Hindernis („Wall attenuation factor“)

Das Modell nimmt also einen logarithmischen Verlust der Signalstärke über die Distanz an. Zudem wird an Hindernissen die Signalstärke linear gedämpft. Dabei wird angenommen, dass die Hindernisse alle aus demselben Material mit derselben Dicke sind.

3.2.2 Multipoint-Relais

Multipoint Relais (MPR, Tc-REs [1995]) sind ein Konzept zur Verbreitung von Nachrichten innerhalb eines mobilen Ad-Hoc Netzwerkes (Mobile Ad-hoc Networks, MANET). Solch ein MANET zeichnet sich dadurch aus, dass die Kommunikationsteilnehmer zusammen ein Netz selbstständig aufbauen und konfigurieren. Dabei ist die Topologie des Netzwerkes vorher nicht bekannt. Dies führt dazu, dass z.B. Kommunikationswege von einem Teilnehmer zu einem anderen Teilnehmer erst gefunden werden müssen.
Innerhalb eines MANET entsteht häufig die Notwendigkeit, jeden Kommunikationsknoten mit einer bestimmten Nachricht zu erreichen. Das Netzwerk muss mit dieser Information „geflutet“ werden. Die naive Verfahrensweise, jeder Knoten sendet die Information weiter bis alle Knoten erreicht sind, birgt die Gefahr einer hohen Überlast. Daher

ist in bestimmten Ad-Hoc-Netzwerkprotokollen das Verfahren des sogenannten „MPR-Flooding“ eingeführt worden [Qayyum et al., 2002]. Es beruht auf der Idee, dass nur eine Teilmenge aller Netzwerkknoten eine Information aussenden muss, um alle Teilnehmer zu erreichen. Die Knoten dieser Teilmenge werden Multipoint Relays (MPRs) genannt. Das Finden der optimalen (und damit kleinsten) Menge an MPRs ist jedoch NP-hart (Viennot [1998]). Allerdings gibt es auch hier Heuristiken, die ein gutes Ergebnis erbringen.
Ein Ansatz ist ein Greedy-Algorithmus. Dabei wird der Sender als erster MPR betrachtet:

- Sei v_i ein noch nicht abgearbeiteter MPR.
- Bestimme die Menge $\mathcal{H}_1$, die alle Nachbarn von v_i enthält.
- Bestimme die Menge $\mathcal{H}_2$, die all die Knoten enthält, die von den Knoten in $\mathcal{H}_1$ erreicht werden können.
- Entferne aus $\mathcal{H}_2$ alle Knoten, die in $\mathcal{H}_1$ enthalten sind oder schon als *erreicht* markiert sind. Damit enthält $\mathcal{H}_2$ alle unerreichten Knoten, die über einen Hop mit v_i verbunden sind.
- So lange $\mathcal{H}_2$ Knoten enthält:
  - Bestimme für jeden Knoten in $\mathcal{H}_1$ die Anzahl c der Knoten aus $\mathcal{H}_2$, die Nachbarn des Knoten sind.
  - Sei v_y der Knoten mit dem größten c.
  - Markiere v_y als *MPR*.
  - Markiere alle Knoten aus $\mathcal{H}_2$, die Nachbarn zu v_y sind, als *erreicht* und entferne sie aus $\mathcal{H}_2$.

Dieser Ansatz kann leicht noch dadurch verbessert werden, dass zuerst alle Knoten in $\mathcal{H}_1$, die einen Knoten aus $\mathcal{H}_2$ alleine erreichen, als MPR gekennzeichnet werden. Diese Knoten werden auf jeden Fall als MPR benötigt und können damit andere MPR Knoten überflüssig machen.

4

Globale Planung unter spatialen Nebenbedingungen

Das in dieser Arbeit vorgestellte Planungsverfahren besteht aus zwei Phasen. Zuerst wird ein sogenannter globaler Mehrroboterplan, der die Nebenbedingungen einhält, für das MRS entwickelt. Solch ein globaler Mehrroboterplan enthält Zielpunkte für die Roboter, sowie Wege zwischen den Zielpunkten. In der zweiten Phase muss dieser globale Mehrroboterplan in Handlungsanweisungen für die einzelnen Roboter zerlegt werden.

In diesem Kapitel wird die erste Phase des Planungsverfahrens beschrieben. Die zugrunde liegende Umwelt, modelliert als 2,5D-Oberfläche, wird für die Planung diskretisiert. Diese Diskretisierung ermöglicht es, verschiedene Informationen in Graphen zu hinterlegen. Dabei handelt es sich zum einen um den Bewegungsgraphen, zum anderen um den Bedingungsgraphen. Diese beiden Graphen zeigen, wo der Roboter hinfahren kann und wo er hinfahren darf. Beide Graphen repräsentieren sehr unterschiedliche Informationen. Eine Verknüpfungsoperation vereinfacht die Planung deutlich (Kapitel 4.1).

Mit der Darstellung des globalen Planungsproblems unter spatialen Nebenbedingungen als Graphenproblem werden zwei Algorithmen vorgestellt, die die globale Planung durchführen können. Ein Algorithmus basiert auf dem Steinerbaum-Problem (Kapitel 4.2.1), während der andere Algorithmus an den MPR-Flooding Algorithmus angelehnt ist (Kapitel 4.2.2).

Zuletzt werden in der Simulation einige exemplarische Beispielplanungen vorgestellt, um einen Eindruck von der Art der erstellten Pläne zu vermitteln.

4.1 Basisgraphen

In der weiteren Arbeit wird die Umgebung diskretisiert und Bereiche, zwischen denen ein Roboter fahren kann, werden verbunden. Dies entspricht einem Roadmap-Verfahren, bei dem ein gleichmäßiges Gitter auf die Umgebung gelegt wird. Anders als in den in der Literatur vorgestellten Verfahren, die Umgebung randomisiert zu samplen, wird eine gleichmäßige Gitterstruktur, auf der die Punkte verteilt werden, gewählt. Diese Punktemenge V bildet die Knoten des Suchgraphen. Jeder dieser Knoten repräsentiert eine Position (bzw. einen kleinen Bereich) in der Umwelt. Die gleichmäßige Gitterstruktur ermöglicht bei der Planung einfach auf bekannte Heuristiken für die Steinerbäume zurückzugreifen.

4.1.1 Bewegungs- und Bedingungsgraph

Der Bewegungsgraph repräsentiert die Bewegungsmöglichkeiten der verwendeten Roboter. Seine Struktur wird sowohl von der Umgebung als auch von den Fähigkeiten des Roboters beeinflusst. Ein Roboter mit Beinen wird andere Hindernisse überwinden können als ein Roboter mit Rädern oder gar ein fliegender Roboter.

Definition 4.1*: Bewegungsgraph*
Seien v_i und v_j zwei unterschiedliche, räumlich benachbarte Knoten aus der Knotenmenge V. Es existiert zwischen v_i und v_j genau dann eine Kante $e \in E_{Bew}$ im Bewegungsgraphen $G_{Bew} = (V, E_{Bew})$, wenn der Roboter von v_i nach v_j auf geradem Weg fahren kann.

Dies bedeutet anschaulich, dass zwischen zwei Knoten, die direkt nebeneinander liegen, genau dann eine Verbindung existiert, wenn in der Umwelt kein Hindernis dazwischen liegt und die Kinematik des Roboters eine solche Bewegung erlaubt. Dabei wird angenommen, dass ein Roboter, der in der Lage ist von v_i nach v_j zu fahren, ebenso in der Lage ist von v_j nach v_i zu gelangen. Der Bewegungsgraph ist daher ungerichtet.
Bei diesem Ansatz ist die Dichte der Knoten, die auf die Umgebung verteilt sind, wichtig. Sind zu wenig Knoten vorhanden, so kann es sein, dass Wege an Engstellen nicht gefunden werden. Existiert nun ein solcher Graph G_{Bew}, so kann ein Pfad von einem Knoten zu einem beliebigen anderen z.B. durch den A*-Algorithmus [Hart et al., 1968] leicht gefunden werden.
Die Besonderheit des hier vorgestellten Planungsverfahrens für Mehrrobotersysteme ist die Berücksichtigung von Nebenbedingungen. Nebenbedingungen sind dabei Bedingungen, die nicht unmittelbar im Zusammenhang mit der Erfüllung der Aufgabe des MRS stehen, jedoch mittelbar erfüllt sein müssen. Als Beispiel sei wiederum die Kommunikation genannt. Wenn die Auswertung der Daten der Roboter in Echtzeit am Leitstand geschehen soll, müssen diese Daten jederzeit dahin gelangen. So wird implizit eine ständige Funkverbindung zwischen den Robotern und dem Leitstand vorausgesetzt.

Definition 4.2*: Spatiale Nebenbedingung*
Seien $\mathbf{x}$ *und* $\mathbf{y}$ *Positionen in der Umwelt. Eine spatiale Nebenbedingung besteht aus zwei Komponenten: einer Abbildung* $\mathcal{C}$ *mit* $\mathcal{C}(\mathbf{x}, \mathbf{y}) = r$ *mit* $\mathcal{C} : \mathbb{R}^3 \times \mathbb{R}^3 \mapsto \mathbb{R}$ *und ein Schwellwert* t*. Gilt* $r \geq t$ *so gilt die Nebenbedingung zwischen* $\mathbf{x}$ *und* $\mathbf{y}$ *als erfüllt, ansonsten als nicht erfüllt.*

Um diese Definition zu verdeutlichen, sollen zwei Beispiele solcher Nebenbedingungen aufgezeigt werden: Zum ersten die sogenannte Sichtbarkeitsnebenbedingung, bei der die Roboter sich untereinander sehen können sollen (siehe Abbildung 4.1). In diesem Fall ist $\mathcal{C}$ definiert als:

$$\mathcal{C}_{vis}(\mathbf{x}, \mathbf{y}) = \begin{cases} 1, & \text{wenn } \mathbf{y} \text{ von } \mathbf{x} \text{ aus zu sehen ist,} \\ 0, & \text{sonst.} \end{cases}$$

Diese Nebenbedingung erzeugt schon einen Wert von entweder Null oder Eins, sodass ein beliebiger Schwellwert $0 < t \leq 1$ gewählt werden kann.

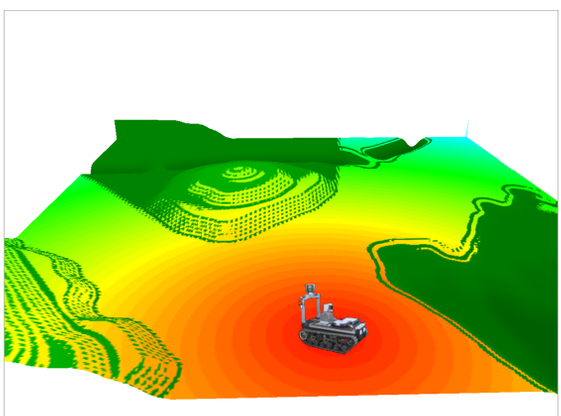


Abbildung 4.1: Sichtbare Bereiche auf einer Oberfläche. Die vom Roboter aus nicht sichtbaren Bereiche sind dunkelgrün.

Ein zweites Beispiel ist die Kommunikations-Nebenbedingung. Durch geeignete Verfahren können Signalstärken vorhergesagt werden. Das Ergebnis kann als ein Hinweis gesehen werden, ob eine Kommunikationsverbindung zwischen zwei Robotern möglich ist. Hierbei ist nun $\mathcal{C}$ definiert als:

$$\mathcal{C}_{sig}(\mathbf{x}, \mathbf{y}) = L,$$

wobei L die vorhergesagte Signalstärke zwischen Punkt $\mathbf{x}$ und Punkt $\mathbf{y}$ ist. Um zu bestimmen, ob die Nebenbedingung zwischen zwei Punkten erfüllt ist, muss festgelegt

werden, ab welcher Signalstärke eine Kommunikation vermutlich möglich sein wird. Diese Signalstärke sei der Schwellwert t. Dies ist z.B. von der verwendeten Hardware oder Protokoll abhängig. Diverse Funkausbreitungsmodelle sind in der Literatur verfügbar und können innerhalb eines Robotersystems auch online angepasst werden (siehe z.B. Brüggemann et al. [2009] und Mostofi et al. [2010]). Eine Visualisierung der Signalstärke in einer beispielhaften Umgebung ist in Bild 4.2 zu sehen.

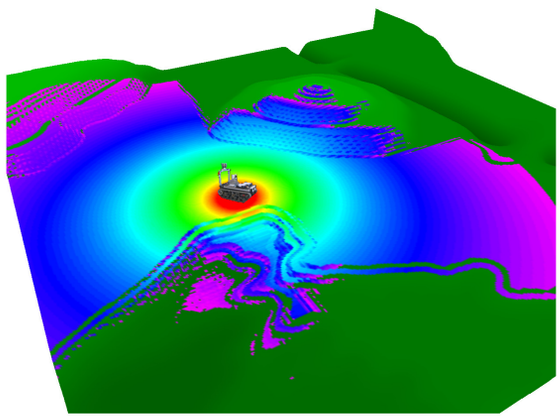


Abbildung 4.2: Berechnete Signalstärke nach dem Log-Distance-Fading-Modell (siehe Goldhirsh and Vogel [1998]). Zusätzlich wird eine Dämpfung bei der Durchdringung der Oberfläche angenommen (solch ein Modell wird z.B. in Bahl and Padmanabhan [2000] beschrieben). Jede eingefärbte Stelle kann mit der Funkverbindung erreicht werden. Hier dürfte also ein anderer Roboter stehen.

Um die Informationen aus den definierten Nebenbedingungen zu repräsentieren, wird der Bedingungsgraph benötigt. Dieser Graph bildet die Nebenbedingung ab, an die sich das MRS zu halten hat, während es sich bewegt. Der Bedingungsgraph hat genau dieselben Knoten wie der Bewegungsgraph, jedoch haben hier die Kanten eine andere Bedeutung:

Definition 4.3*: Bedingungsgraph zu einem Bewegungsgraphen $G_{Bew} = (V, E_{Bew})$*
Seien v_i und v_j zwei unterschiedliche Knoten aus der Menge aller Knoten V. Im Bedingungsgraphen G_{Bed} existiert zwischen v_i und v_j genau dann eine Kante, wenn $\mathcal{C}(v_i, v_j)$ größer als ein bestimmter Wert ist. Hierbei definiert $\mathcal{C}$ die Nebenbedingung. Ist $\mathcal{C}$ nicht symmetrisch, also $\mathcal{C}(v_i, v_j) \neq \mathcal{C}(v_j, v_i)$, so ist G_{Bed} ein gerichteter Graph.

Dies bedeutet, dass genau dann im Bedingungsgraph eine Kante zwischen zwei Punkten in der Umwelt existiert, wenn zwischen diesen beiden Punkten die Nebenbedingung erfüllt ist. Solch eine Nebenbedingung ist also immer zwischen zwei Punkten definiert (siehe 4.3).

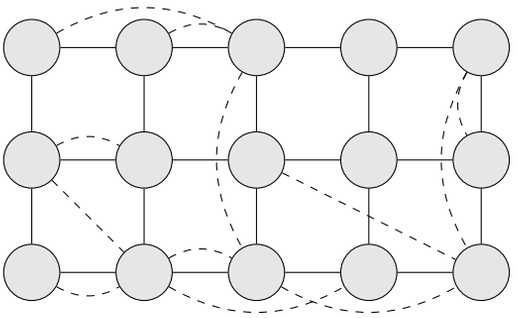


Abbildung 4.3: Schematische Darstellung des Bewegungs- und Bedingungsgraphen, hier ohne Diagonalen im Bewegungsgraph. Die Kanten des Bewegungsgraphen sind durchgezogen, die Kanten des Bedingungsgraphen sind gestrichelt.

Die als Beispiel gezeigten Nebenbedingungen sind alle symmetrisch, das bedeutet die resultierenden Bedingungsgraphen sind ungerichtet. Im Folgenden wird für alle Nebenbedingungen angenommen, dass sie einen ungerichteten Bedingungsgraph erzeugen. Dies gilt für alle in der Literatur gefundene Nebenbedingungen, die für Roboter angenommen wurden, wie z.B. die Berechnung der Kommunikationsreichweite in Mosteo et al. [2008] und Jnaneshwar et al. [2009].
Innerhalb des Bedingungsgraphen gilt, dass die Kantengewichte alle gleich 1 sind. Da in dieser Arbeit nur boolesche Nebenbedingungen betrachtet werden, gibt es keinen quantitativen Unterschied zwischen Kanten im Bedingungsgraph.
Im Bewegungsgraph werden Kanten zwischen horizontalen und vertikalen Nachbarn mit dem Gewicht 1 definiert. Dies entspricht einem Schritt. Um die größere Distanz zu diagonalen Nachbarn darzustellen und „Zick-Zack“ Bewegungen in der Planung zu vermeiden, werden die Kanten zu den diagonalen Nachbarn mit einem Gewicht von $\sqrt{2}$ belegt.

4.1.2 Verknüfungsoperation von G_{Bew} und G_{Bed}

Bei der Bahnplanung unter Nebenbedingung ist es notwendig, die beiden unterliegenden Graphen (Bewegungs- und Bedingungsgraphen) häufig miteinander abzugleichen. Dies ergibt sich daraus, dass der Bewegungsgraph die möglichen Optionen zur Bewegung des Roboters darstellt, der Bedingungsgraph allerdings anzeigt, ob die Bewegung derzeit erlaubt ist. Um diesen Abgleich zwischen den beiden Graphen schnell durchführen zu können, wird eine Graphenoperation auf den beiden Graphen eingeführt. Diese Graphenoperation erzeugt den sogenannten Separated Connection Graph (SCG). Die Verknüpfungsoperation wird über einen bestimmten Knoten definiert. So ist ein SCG immer einem bestimmten Knoten v zugeordnet. Dabei ist SCG(v) wieder ein Graph.

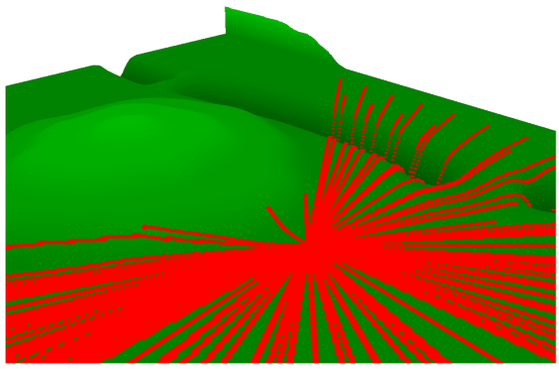


Abbildung 4.4: Ausschnitt des Sichtbarkeits-Bedingungsgraph für einen Knoten. Der Hügel schränkt die Sicht in eine Richtung ein, der Graben dagegen nicht.

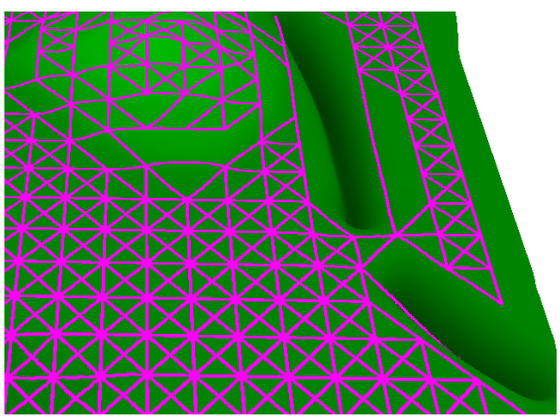


Abbildung 4.5: Ausschnitt des Bewegungsgraphen auf einer 3D Oberfläche. Die Möglichkeit der Bewegung wird durch die Steigung eingeschränkt. Zu steile Passagen am Hügel können nicht befahren werden. Ebenso bilden die Gräben ein unüberwindliches Hindernis.

Definition 4.4*: Separated Connection Graph(v)*
Seien $G_{Bew} = (V, E_{Bew})$ und $G_{Bed} = (V, E_{Bed})$ der Bewegungs- bzw. Bedingungsgraph. Für jeden Knoten $v \in V$ ist der Separated Distance Graph SCG(v) $= (V_S, E_S)$ definiert als die Verknüpfung beider Graphen am Knoten v. Dabei enthält $V_S \subseteq V$ den Knoten v und seine direkten Nachbarn in G_{Bed}. Eine Kante $e(v, v_j)$ mit $v_j \in V_S$ ist dann in E_S enthalten, wenn gilt: Es existiert ein Pfad Π in G_{Bew} von v nach v_j, dessen Knoten alle in V_S liegen.

Was ein SCG anschaulich leistet, kann in Abbildung 4.7 nachvollzogen werden. Das SCG(v) kann als „Look up table“ für die erreichbaren Knoten betrachtet werden. Steht ein Roboter auf dem Knoten v, so können alle Knoten, die in SCG(v) mit v verbunden sind, angefahren werden, ohne dass die Nebenbedingung verletzt wird. Weitere Eigen-

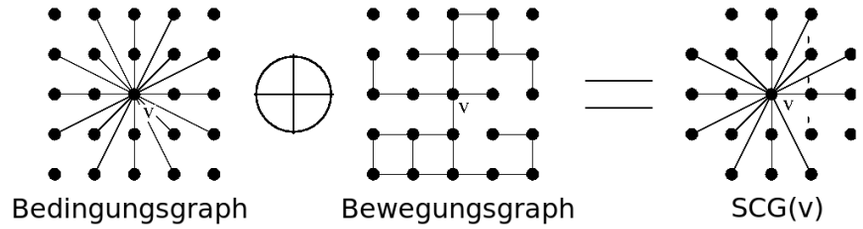


Abbildung 4.6: Die Verknüpfung von G_{Bew} und G_{Bed}. Der linke Operand ist der Bedingungsgraph, der rechte der Bewegungsgraph. Der Bedingungsgraph definiert, welche Knoten betrachtet werden; der Bewegungsgraph, welche verbunden werden. In diesem Fall sind bestimmte Knoten durch den Wegfall der Eckknoten nicht mehr erreichbar.

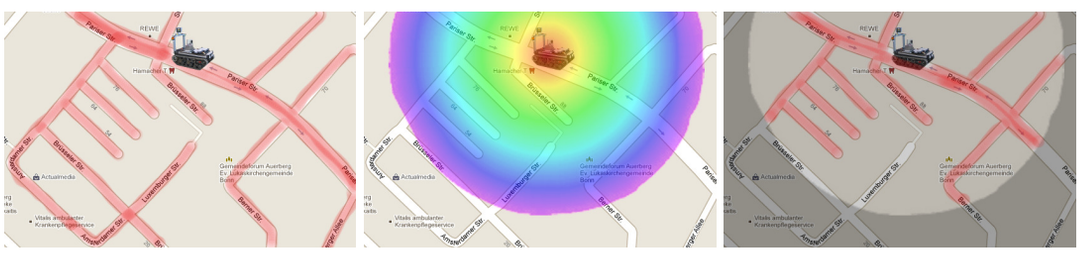


Abbildung 4.7: Anschauliche Darstellung eines SCGs. Das linke Bild zeigt alle die Straßen, die der Roboter befahren kann (analog zu einem Bewegungsgraphen). Das mittlere Bild zeigt die mögliche Funkreichweite (analog zu dem Bedingungsgraphen). Das rechte Bild zeigt die Kombination aus den beiden linken. Alle befahrbaren Straßen, die in der Funkabdeckung erreichbar sind, sind rot markiert (analog zum SCG).

schaften des SCGs wie die Vollständigkeit der direkten Wege und der Asymmetrie der SCGs werden in Anhang B.1 näher betrachtet und bewiesen.

Komplexität der SCG-Berechnung

Zuerst müssen die direkten Nachbarn von v in G_{Bed} bestimmt werden. Mit Hilfe einer geeigneten Datenstruktur für den Graphen (z.B. Adjazenzliste) kann dies in O(n) geschehen, wobei n die Anzahl der Knoten im Bewegungs- bzw. Bedingungsgraphen ist. Um zu überprüfen, ob ein Pfad Π wie in der Definition gefordert existiert, wird ein Teilgraph $G_{sub} \subseteq G_{Bew}$ erzeugt, der nur die Knoten V_S und entsprechende Kanten aus dem Bewegungsgraph besitzt (in O(n)). In G_{sub} wird die Zusammenhangkomponente, in der v liegt, bestimmt (ebenfalls in O(n) mittels Tiefensuche). Für alle Knoten, die in dieser Zusammenhangkomponente liegen, existiert ein solcher Pfad Π. Damit können SCGs in folgender Zeitkomplexität erzeugt werden:

	SCG(v)
Direkte Nachbarn von v_i	O(n)
Erzeugen von G_{sub}	O(n)
Zusammenhangskomponente von v in G_{sub}	O(n)
Komplexität	O(n)

SCG-Metagraph

Der SCG-Metagraph ist ein Graph, der die Verknüpfung aller möglichen SCGs von Bedingungs- und Bewegungsgraph darstellt. In ihm repräsentieren die Knoten die einzelnen SCGs und ermöglichen dadurch eine Planung, die sowohl Bewegungs- als auch Bedingungsgraph berücksichtigt.

Definition 4.5*: SCG-Metagraph*
Der SCG-Metagraph $M_{scg} = (V, E_{scg})$ ist die Vereinigung aller SCG(v) für $v \in V$. Es existiert genau dann eine gerichtete Kante $e(v_i, v_j) \in E_{scg}$ mit $v_i, v_j \in V$, wenn v_j in SCG(v_i) mit v_i verbunden ist.

Da „Verbindung“ zwischen $v_i, v_j \in V$ zwischen v_i und v_j in SCG(v_i) nicht bedeutet, dass es dieselbe Verbindung in SCG(v_i) gibt (Beweis siehe Anhang B.5), ist M_{scg} im Gegensatz zu den bisher verwendeten Graphen immer **gerichtet**.
Bis auf die Tatsache, dass der SCG-Metagraph im Aufbau aufwändig ist, eignet er sich, Wege zu finden, die die Nebenbedingung nicht verletzen. So kann der Weg von v_i nach v_j z.B. durch den Dijkstra Algorithmus gefunden werden. Dabei bilden die Knoten, über die der Pfad verläuft, Positionen für permanente Relais-Roboter.

4.1.3 Begriffsdefinitionen

Mit den Annahmen über die Umgebung und den definierten Graphen kann nun eine Instanz des Problems der Navigation unter Nebenbedingungen, wie es in der Motivation geschildert wurde, beschrieben werden:

Definition 4.6*: Positionierung*
In einer Positionierung P ist jedem Roboter $\mathcal{A}_i$ aus dem MRS eine Position und damit ein Knoten v_i zugewiesen.

Bei der Definition der Position ist zudem explizit zu erwähnen, dass auch mehrere Roboter einem Knoten zugewiesen werden können. Dann wird angenommen, dass die Roboter so eng zusammenstehen, dass es insbesondere für die Nebenbedingung keinen Unterschied mehr darstellt, welcher Roboter betrachtet wird.

Definition 4.7*: Gültige Positionierung*
Eine Positionierung ist genau dann eine gültige Positionierung, wenn der Teilgraph G_{Bed}^{sub} aus G_{Bed}, der genau die Knoten enthält, auf denen Roboter stehen, aus einer Zusammenhangskomponente besteht.

Dies bedeutet, dass, wenn die Roboter eine gültige Positionierung eingenommen haben, die Nebenbedingung für das gesamte MRS erfüllt ist.
Zusätzlich wird der Begriff der Bewegung für Bewegungsgraphen definiert:

Definition 4.8*: Bewegung*
Seien P_i mit $i = 1 \dots n$ Positionierungen. Dann ist der Übergang von P_j nach P_{j+1} für $1 \leq j \leq n-1$ eine Bewegung, wenn genau ein Roboter $\mathcal{A}_i \in \mathcal{A}$ ein anderer Knoten zugewiesen wurde, nämlich v_j in P_j bzw. v_{j+1} in P_{j+1}. Zudem muss gelten das $v_j \neq v_{j+1}$ und $\exists e(v_j, v_{j+1}) \in E_{Bew}$.

Weiterhin werden verschiedene spezielle Konstrukte innerhalb des Problemkreises der koordinierten Navigation von Mehrrobotersystemen unter Nebenbedingungen angesprochen. Um diese eindeutig nutzen zu können, werden ihre Eigenschaften hier definiert.
Bedingt dadurch, dass durch die Nebenbedingungen für einen Roboter nicht immer der direkte Weg befahrbar ist, werden weitere Roboter benötigt, um die Nebenbedingung zu erfüllen. Solche Roboter werden Relais-Roboter genannt.

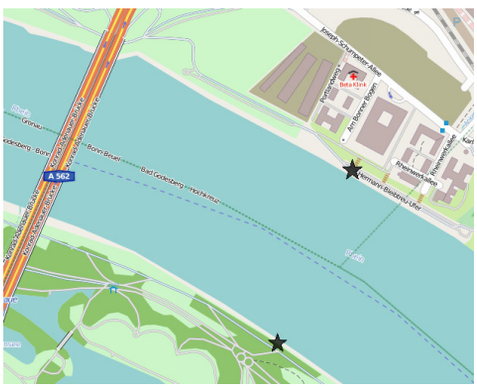
Definition 4.9 *Relais-Roboter (RR)*
Ein Relais-Roboter $\mathcal{A}_r$ ist ein Roboter, der auf einem Knoten abgestellt wird, damit ein anderer Roboter $\mathcal{A}_i$ einen Zielknoten erreichen kann, ohne dass $\mathcal{A}_i$ die Nebenbedingung verletzt.

Definition 4.10 *Permanenter Relais-Roboter (PRR)*
Ein permanenter Relais-Roboter $\mathcal{A}_{pr}$ ist ein Relais-Roboter, der, nachdem $\mathcal{A}_i$ sein Ziel erreicht hat, nicht mehr bewegt werden darf, da sonst die Nebenbedingung verletzt wird.

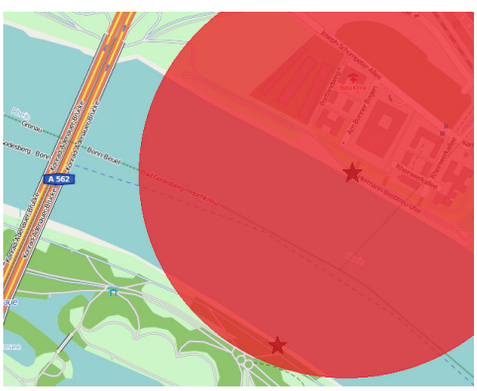
Auch wenn der Zielpunkt für einen Roboter selber die Nebenbedingung im Gesamtsystem erfüllt, kann, bedingt durch die Umgebung, der Weg dahin eine Verletzung der Nebenbedingung bedeuten. Um dies zu verhindern, wird auch ein Relais-Roboter benötigt. Allerdings kann dieser Relais-Roboter, nachdem der andere Roboter zum Ziel gelangt ist, für andere Aufgaben verwendet werden. Deswegen benötigt man ihn nur temporär.

Definition 4.11 *Temporärer Relais-Roboter (TRR)*
Ein temporärer Relais-Roboter $\mathcal{A}_{tr}$ ist ein Relais-Roboter, der, nachdem $\mathcal{A}_i$ seinen Zielknoten erreicht hat, abgezogen werden kann, ohne dass die Nebenbedingung verletzt wird (siehe Abbildung 4.8).

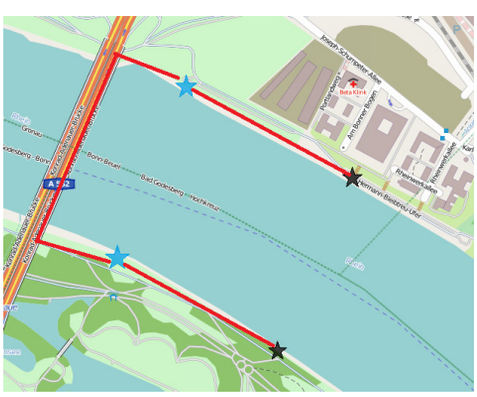
Die Nutzung von Relais-Robotern bedeutet immer sowohl einen Umweg für das Gesamtsystem als auch die Notwendigkeit eines zusätzlichen Roboters. Daher ist das Fin-



Beispiel für die Notwendigkeit von TRRs. Im Bild links sind zwei Positionen für PRR definiert worden (schwarze Sterne).



Dabei sind die Positionen so ausgewählt, dass die Nebenbedingung (roter Kreis) erfüllt ist. Die Brücke als einzige Verbindung zwischen den beiden Positionen ist jedoch nicht abgedeckt.



Um von einer Position zur anderen zu gelangen, müssen zusätzliche Relais-Roboter eingesetzt werden (blaue Sterne). Diese ermöglichen den Weg von einem PRR zum anderen, ohne die Nebenbedingung zu verletzten. Nachdem auf beiden PRR-Positionen ein Roboter steht, können die Roboter auf den TRR-Positionen abgezogen werden, ohne die Nebenbedingung zu verletzten.

Abbildung 4.8: Beispiel für die Anwendung von TRRs

den eines direkten Weges von Vorteil.

Definition 4.12 *Direkter Weg*
Steht ein Roboter auf v_i, dann existiert genau dann ein direkter Weg zwischen v_i und v_j, wenn ein anderer Roboter von v_i nach v_j gelangen kann, ohne einen Relais-Roboter zu benötigen.

4.1.4 Navigation unter Nebenbedingungen als graphentheoretisches Problem

Mit Hilfe der definierten Darstellung der Umgebung kann nun eine Instanz des koordinierten Navigationsproblems formal beschrieben werden. Diese Formalisierung des Problems bildet die Basis für die weitere Herangehensweise. Die hier gewählte Darstellung beeinflusst dabei sowohl die weitere Vorgehensweise wie auch die gefundenen Lösungen in erheblichem Maße. Insbesondere die aus dem Roadmapverfahren entstehenden Graphen führen zu einer Diskretisierung der Umgebung, welche die Lösungen bestimmt.
Formal kann das Problem der koordinierten Navigation unter spatialen Nebenbedingungen nun wie folgt beschrieben werden:
Gegeben ist:

- Eine Umwelt (z.B. als Höhenmodell)
- Eine Nebenbedingung
- V als eine Menge von Knoten in der Umwelt
- $\mathcal{A}$ als die Menge aller Roboter
- $\mathcal{Z}$ als eine Untermenge von V. Dies sind die Ziele der Roboter. $|\mathcal{Z}|$ kann nicht größer sein als die Anzahl Roboter.

Dann sind G_{Bew} und G_{Bed} der Bewegungs- bzw. der Bedingungsgraph wie oben definiert und S_{start} die gültige Positionierung zu Beginn.
Nun kann das Koordinierungsproblem in zwei Teile geteilt werden:

1. Gesucht: Die Endpositionierung
   Suche eine gültige Positionierung, die mindestens alle Ziele enthält. Diese gültige Positionierung heißt Endpositionierung S_{end}.
2. Gesucht: Bewegung zur Endpositionierung
   Suche eine Kette von gültigen Positionierungen $\{S_{start}, S_1, S_2, \ldots, S_{end-1}, S_{end}\}$, wobei S_t durch eine Bewegung aus S_{t-1} hervorgegangen ist.

Innerhalb der vorliegenden Arbeit wird eine weitere Einschränkung des Problems angenommen: die Roboter starten alle von einem Punkt aus. Dies ist typisch für die betrachtete Anwendung, da die Roboter üblicherweise von einem Punkt aus eingesetzt werden, und entspricht einer gültigen Positionierung nach der Definition. Sollten die Roboter

zu Beginn zwar auf einer gültigen Position, jedoch nicht zusammen stehen, ergibt sich eine andere Problemstellung. Hier müssten die Roboter entweder zunächst zum Startpunkt zusammengezogen werden oder, ohne die Nebenbedingung zu verletzten, auf den berechneten globalen Mehrroboterplan bewegt werden. Dieser Aspekt wird in dieser Arbeit nicht weiter behandelt, wenngleich Teile davon im Kapitel 6 betrachtet werden, um von einem globalen Mehrroboterplan auf einen anderen, aktualisierten Plan zu wechseln.

4.2 Finden einer Endpositionierung

Wie oben vorgeschlagen wird das Problem der koordinierten Navigation unter Nebenbedingungen in zwei Problemteile zerlegt. Der erste Teil besteht dabei aus dem Problem, eine sogenannte Endpositionierung zu finden. Dabei handelt es sich um eine Positionierung, die die Nebenbedingung erfüllt und zudem alle Zielpunkte beinhaltet.

In diesem Unterkapitel werden zwei grundsätzlich verschiedene Algorithmen zur Suche einer solchen Endpositionierung entwickelt. Der Steinerbaum-Planer (STPlaner) betrachtet dabei das Problem unabhängig von der Tatsache, dass Roboter diesen Plan ausführen sollen. Er nutzt die Äquivalenz zwischen dem Finden einer Endpositionierung und eines Steinerbaums aus. Damit liefert der Steinerbaum-Planer Endpositionierungen , die nur wenige Roboter benötigen. Es wird jedoch gezeigt, dass in bestimmten Umgebungen (z.B. mit Abbruchkanten) solche minimalen Endpositionierungen nur äußerst schwer und mit viel Aufwand von den Robotern zu erreichen sind.

Der AgentPlaner als zweiter Algorithmus nutzt Informationen über die verwendeten Roboter, um eine zum Steinerbaum-Planer qualitativ verschiedene Endpositionierung zu erzeugen. Er nutzt zusätzlich zum Bedingungsgraph auch noch den Bewegungsgraph, der Informationen darüber enthält, wie sich die Roboter im Mehrrobotersystem bewegen können. Dabei werden in einer lokalen Suche nur solche Knoten betrachtet, die sowohl die Nebenbedingung einhalten, wie auch einfach von einem Roboter zu erreichen sind. So ergeben sich beim AgentPlaner Endpositionierungen, die zwar geringfügig mehr Roboter benötigten als beim STPlaner, allerdings sind solche Endpositionierungen deutlich einfacher von einem Robotersystem einzunehmen.

4.2.1 Steinerbaum-Planer

In diesem Abschnitt wird zunächst der Zusammenhang zwischen dem Finden einer Endpositionierung und Steinerbäumen gezeigt. Dieser Zusammenhang wird genutzt, um den Algorithmus STPlaner zu entwickeln.

Endpositionierung und Steinerbäume

Die Endpositionierung ist, wie oben beschrieben, eine gültige Positionierung, die den Startpunkt sowie alle Endpunkte enthält. Damit entspricht die Suche nach einer gültigen Endpositionierung der Suche nach einem Graphen G^{sub}_{Bed}, in dem der Startpunkt sowie die Zielpunkte in derselben Zusammenhangskomponente liegen. Diese Einschränkung alleine führt jedoch zu vielen unterschiedlichen Lösungen. So könnte mit einer einfachen Tiefensuche die Zusammenhangskomponente G^{sub}_{Bed} in G_{Bed}, in der der Startpunkt und die Zielpunkte liegen, bestimmt werden. Dies wäre eine gültige Endpositionierung. So könnten auch weitere Endpositionierung generiert werden, indem aus G^{sub}_{Bed} Knoten entfernt werden, die nicht Start- oder Zielpunkt sind und die G^{sub}_{Bed} nicht in mehrere Zusammenhangskomponenten zerfallen lassen. Solche Lösungen sind zwar gültig aber nicht sinnvoll, da sie im Allgemeinen viel zu viele permanente Relais-Roboter enthalten. Daher ist zu überlegen, welche zusätzlichen Bedingungen an die Endpositionierung zu stellen sind, damit sie auch im praktischen Einsatz sinnvoll umgesetzt werden kann.
Bei dieser Art Pfadplanung sind drei Arten von Gütemaßen üblich. So kann die Endpositionierung daraufhin optimiert werden, dass der kleinste Gesamtweg entsteht. Dies bedeutet, der Weg, der zurückzulegen ist, um die resultierende Endpositionierung (die einen Baum darstellt) zu erreichen, hat insgesamt eine minimale Länge. Eine weitere Metrik stellt die Minimierung des längsten Weges in der Endpositionierung dar. Da dieser Weg vermutlich auch die meiste Zeit benötigt, um abgefahren zu werden, wird hier indirekt die Ausführungszeit optimiert. Als dritter möglicher Optimierungsweg bietet sich die Frage nach der Anzahl der benötigten Roboter an. In einem Mehrrobotersystem ist üblicherweise die Anzahl der verfügbaren Roboter ein stark limitierender Faktor. Insbesondere aus Kostengründen kann die Anzahl der Roboter nicht beliebig angehoben werden. Daher wird in dieser Arbeit die Reduzierung der Anzahl der Roboter als vordringliches Ziel bei der Erstellung der Pläne angesehen.
Somit kann das Finden einer Endpositionierung folgendermaßen gesehen werden: Gesucht wird ein Teilgraph G_{st} des Bedingungsgraphen, der sowohl die Zielpositionen als auch die Startposition beinhaltet und möglichst wenige Knoten enthält. Dabei soll die Anzahl der Knoten minimiert werden, da jeder Knoten in der Endpositionierung einen permanenten Relaisroboter (PRR) repräsentiert. Da im Bedingungsgraph die Kantengewichte gleich eins sind, entspricht damit die Suche nach der minimalen Anzahl Knoten der Suche nach einem Teilgraph mit minimalem Kantengewicht. Dabei ergibt sich implizit eine Baumstruktur für den Teilgraphen. Wäre G_{st} kein Baum, so gäbe es einen Kreis in diesem Graphen. Das heißt es gäbe mindestens eine Kante, die noch entfernt werden könnte, ohne dass der Graph zerfallen würde. Also war G_{st} noch nicht minimal.
Mit der Betrachtung von Ziel- und Startpunkten als Terminal entspricht dies genau dem Steinerbaum-Problem (siehe Gilbert and Pollak [1968]). Dies ist ein bekanntes Problem aus der Algorithmischen Geometrie. In der praktischen Anwendung wird es hauptsächlich in der Kommunikation und in der technischen Informatik, namentlich beim Netzwerkrouting (z.B. in Waxman [1988]; Lim and Kim [2000] und vielen an-

deren) und im Chipdesign (z.B. bei Chiang et al. [1990]; Hong et al. [1993]; Xu et al. [2003] und vielen anderen) angewandt.

Lemma 4.13 *Die Lösung des Steinerbaum-Problems auf dem Bedingungsgraphen ergibt eine gültige Endpositionierung mit der minimalen Anzahl an Robotern.*

Um Lemma 4.13 beweisen zu können, wird gezeigt, dass die Lösung des Steinerbaumes eine gültige Endpositionierung ist.

Lemma 4.14 *Seien der Startpunkt und alle Zielpunkte die Terminale für das Steinerbaum-Problem. Gesucht wird der Steinerbaum auf dem Graphen G_{Bed}. Der resultierende Steinerbaum G_{st} ist eine gültige Endpositionierung.*

Beweis. Eine gültige Endpositionierung muss auf dem Graphen G_{Bed} zusammenhängend sein. Diese Bedingung erfüllt der Steinerbaum durch seine Baumstruktur. Zusätzlich müssen sowohl Start- als auch die Zielpunkte enthalten sein. Dies wird durch die Wahl der Terminale sichergestellt. ■

Nun kann Lemma 4.13 gezeigt werden.

Beweis. Der resultierende Steinerbaum ist nach Lemma 4.14 eine gültige Endpositionierung. Da der Steinerbaum ein Teilgraph des Bedingungsgraphen ist, sind alle Kantengewichte gleich eins. Somit ist das Gewicht des Steinerbaums gleich der Anzahl der Kanten. In einem Baum ist die Anzahl der Kanten gleich der Anzahl der Knoten plus eins. Da der Steinerbaum das Gesamtgewicht minimiert, minimiert er die Anzahl der Knoten und damit die Anzahl der Roboter. ■

So kann also mit Hilfe des Steinerbaums eine Endpositionierung mit minimaler Anzahl an Robotern gefunden werden. Es ist bekannt, dass das Steinerbaum-Problem NP-vollständig ist. Hierzu sind diverse Heuristiken bekannt, die in polynomialer Zeit einen nahezu optimalen Steinerbaum berechnen.
Mit der Betrachtung einer Endpositionierung als Steinerbaum können einige Eigenschaften einer guten Endpositon beschrieben werden:

Lemma 4.15 *Es werden mindestens so viele Roboter benötigt wie der Steinerbaum Knoten hat.*

Beweis. Der Beweis ergibt sich direkt aus der Optimalität des Steinerbaums in Bezug auf die Anzahl der Roboter. Der errechnete Steinerbaum ist eine Endpositionierung. Da er die Anzahl Roboter minimiert, existiert auch keine Endpositionierung mit weniger Robotern. ■

Dies ist eine harte untere Grenze. Allerdings gibt es bei beliebigen Nebenbedingungen keine obere Schranke für die benötigte Anzahl von Robotern.

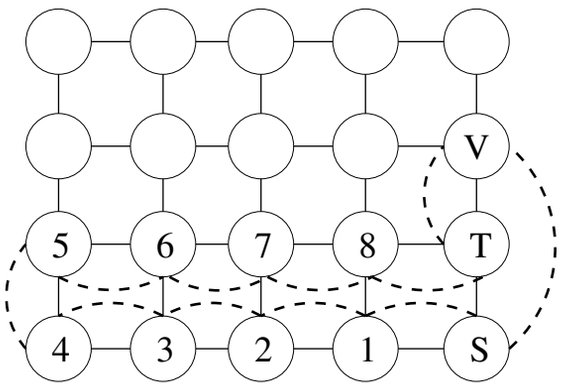


Abbildung 4.9: Es können beliebig viele Roboter benötigt werden, um zu einem Knoten zu gelangen. Die gestrichelten Linien symbolisieren die Kanten aus dem Bedingungsgraph, die durchgezogenen Linien die Kanten des Bewegungsgraphes. Ein Roboter, der von S nach V gelangen will, benötigt auf allen nummerierten Knoten temporäre Relaisroboter.

Lemma 4.16 *Bei einer beliebigen Nebenbedingung nach Definition gibt es keine obere Grenze dazu, wie viele Roboter benötigt werden, um eine Position zu erreichen, bis auf die triviale Grenze. Die triviale Grenze ist die Gesamtanzahl an vorhandenen Knoten.*

Idee. In Abbildung 4.9 ist solch ein Fall zu sehen. Kein Roboter kann von S (Start) direkt zu seinem Ziel (V) gelangen. Angenommen auf S steht ein weiterer Roboter. Sobald nun der erste Roboter auf direktem Wege von S nach V T betritt, ist die Nebenbedingung verletzt (der Teilgraph zerfällt in zwei Teile). Es ist tatsächlich so, dass der Roboter von S aus nur auf den Knoten 1 vorrücken kann, ohne die Nebenbedingung zu verletzten. Hier muss er stehen bleiben. Dann kann aber ein weiterer Roboter auf den Knoten 2 wechseln (von $S \rightarrow 1 \rightarrow 2$). Diese Kette kann so weitergeführt werden, bis ein Roboter auf T steht. Von hier kann er ohne die Nebenbedingung zu verletzten, nach V wechseln. Es ist leicht einzusehen, dass solch eine Kette beliebig verlängert werden kann.

Dieses Konstrukt zeigt auch ein weiteres Problem, welches bei der Planung zur Erreichung der Zielposition auftreten kann. Die Endpositionierung wird nur nach der Betrachtung des Bedingungsgraphen berechnet. Der Roboter muss sich aber entsprechend des Bewegungsgraphen bewegen. Daher müssen beide Graphen zusammengebracht werden. Wie Abbildung 4.9 zeigt, muss ein Weg zwischen zwei Knoten nicht notwendiger Weise direkt sein. Ebenso können mehrere temporäre Relais-Roboter benötigt werden.

STPlaner auf unterschiedlichen Graphen

Die Gültigkeit einer Endpositionierung ist nur auf dem Bedingungsgraphen definiert. Die Endpositionierung entspricht einem Steinerbaum auf dem Bedingungungsgraphen mit dem Startpunkt und den Zielpunkten als Terminale. Folglich ist der triviale Ansatz, den gesamten Bedingungsgraphen zu betrachten. Dabei entsteht das Problem, dass in die Endpositionierung Knoten aufgenommen werden können, die nachher vom Startknoten aus nicht erreichbar sind. Ein erster Schritt, solch einen Fall zu verhindern, kann dadurch erreicht werden, dass im Bewegungsgraph nur die Knoten betrachtet werden, die in einer Zusammenhangskomponente mit dem Startknoten liegen. Der Bedingungsgraph wird nun auch auf diese Knoten reduziert. Somit sind Knoten, die definitiv nicht erreicht werden können, von der Planung ausgeschlossen. Dies reicht jedoch nicht aus, um zu garantieren, dass die Lösung auch erreichbar ist.
Eine solche Garantie kann mit Hilfe des SCG-Metagraph gegeben werden. Ist der SCG-Metagraph berechnet, so können die Knoten im Bedingungsgraph auf diejenigen Knoten beschränkt werden, die im SCG-Metagraph in der Zusammenhangskomponente des Startknotens liegen. Aufgrund des Aufbaus der SCGs und des Metagraphens ist dann eine Lösbarkeit garantiert.
Ist jedoch schon eine Endpositionierung berechnet, kann im Allgemeinen auch ohne die Berechnung des gesamten SCG-Metagraphs getestet werden, ob die Endpositionierung erreicht werden kann. Dabei wird zwischen Start- und jedem Endpunkt jeweils eine Kette von SCGs gesucht. Gibt es jede dieser Ketten, so existiert mindestens eine Lösung für diese Endpositionierung. Dies entspricht dem Test auf die Zusammenhangskomponente im SCG-Metagraphen. Im schlechtesten Fall ist dieser Test dabei so aufwändig wie der Aufbau des SCG-Metagraphen. In der Regel ist jedoch deutlich weniger Aufwand zu erwarten.

Laufzeitanalyse

Der Algorithmus STPlaner wurde vornehmlich dafür entwickelt eine „Baseline“ für das Navigationsproblem unter Nebenbedingungen zu erhalten. Dafür wurde so weit möglich auf bekannte Graphenalgorithmen zurückgegriffen. In diesem Unterkapitel soll nun die Worst Case Laufzeit des STPlaner gezeigtwerden.
Der Algorithmus STPlaner kann in zwei Teile aufgeteilt werden: die Vorberechnungen und die Anfrage (Query). Innerhalb der Vorberechnung werden der Bewegungsgraph und der Bedingungsgraph aufgebaut.
Damit entsteht für das Preprocessing folgende Rechenkomplexität (sei n die Anzahl der Knoten im Bedingungsgraphen):

- Aufbau G_{Bew}: $O(n)$
- Aufbau G_{Bed}: $O(\text{Komplexität der Bedingung} \cdot n^2)$

Zur Erklärung:

- Bewegungsgraph: Für jeden Knoten werden die direkten Nachbarn betrachtet. Die Anzahl der Nachbarn ist konstant für jeden Knoten (≤ 8). Daher kann der Bewegungsgraph in linearer Zeit in Bezug auf die Anzahl der Knoten berechnet werden.
- Bedingungsgraph: Beim Aufbau des Bedingungsgraphen wird jeder Knoten mit jedem anderen Knoten in Beziehung gesetzt. Daher entsteht eine quadratische Laufzeit in Bezug auf die Anzahl der Knoten. Im Gegensatz zum Bewegungsgraphen sollte beim Bedingungsgraphen der Vorfaktor jedoch nicht außer Acht gelassen werden, da er, je nach Nebenbedingung, die Laufzeit entscheidend prägt. So benötigt z.B. die Distanzbedingung nur eine einfache Berechnung der Distanz zwischen zwei Knoten. Bei der Sichtbarkeitsnebenbedingung muss die Sichtverbindung der zwei Knoten überprüft werden, wobei eine Strahlenverfolgung (raytracing) in der Umgebung genutzt wird. Komplexe Wellenausbreitungsmodelle für die Kommunikations-Nebenbedingung können noch rechenintensiver sein, da z.B. Schnitttests in der Umgebung und mehrfache Strahlenverfolgung (zur Simulation der Mehrwegeausbreitung eines Signals) durchgeführt werden müssen.

Für eine Anfrage muss dann ein Steinerbaum auf dem Bedingungsgraphen berechnet werden. Die gewählte Steinerbaum-Heuristik ist der Algorithmus von Mehlhorn Mehlhorn [1988]. Dieser Algorithmus ist 2-kompetitiv und hat eine Komplexität von $O(e + n \cdot log(n))$, wobei e die Anzahl der Kanten und n die Anzahl der Knoten ist.

4.2.2 Agenten-basierter Planer

In vorhergehenden Abschnitt wurden Eigenschaften des Navigationsproblems gezeigt und ein Algorithmus vorgeschlagen, der auf der Lösung des Steinerbaum-Problem basiert. Diese Lösung hat den Vorteil, theoretisch gut analysierbar zu sein. Allerdings berücksichtigt der STPlaner nicht, dass die entstehende Endpositionierung für ein Mehrrobotersystem erstellt wird.
In seiner ursprünglichen Form wird innerhalb des STPlaner nur der Bedingungsgraph bzw. ein Teilgraph genutzt. Dadurch entstehen Endpositionierungen, die nur wenige Roboter benötigen. Schon in den Simulationen zeigt sich jedoch, dass diese Endpositionierungen oft schwer zu erreichen sind und zudem viele temporäre Relais-Roboter (TRR) benötigen. Neben der Problematik der Erreichbarkeit zeigt sich beim STPlaner auch das Problem, dass bei dynamischen, d.h. veränderlichen Umgebungen nach jeder detektierten Änderung der Steinerbaum neu berechnet werden muss.
Für die Pfadplanung eines MRS ist es sinnvoll, die Informationen aus Bewegungsgraph und Bedingungsgraph zusammenzuführen. Um zudem später dynamische Umgebungen behandeln zu können, soll die Suche nach gültigen Endpositionierung mit Hilfe eines Verfahrens durchgeführt werden, bei dem die einzelnen Elemente als eigenständige Agenten betrachtet werden, die miteinander kommunizieren können. Dabei soll der Algorithmus implizit auf veränderliche Umgebungen angewendet werden können, ohne dass der eigentliche Algorithmus für statische Lösungen geändert werden muss.

Schlussendlich bildet dieser Ansatz nur eine weitere Möglichkeit, das Steinerbaum-Problem approximativ zu lösen, da auch dieser Algorithmus eine gültige Endpositionierung in Form eines Baumes berechnet. Da sowohl der Bewegungsgraph wie auch der Bedingungsgraph berücksichtigt werden, kann die Endpositionierung auch die Bewegungseigenschaften des Mehrrobotersystems einbeziehen.
Auch bei diesem Ansatz wird auf viele Eigenschaften und Datenstrukturen aus dem vorherigen Kapitel aufgebaut. Die Agentenplanung ist dabei nur eine andere Möglichkeit, die Endpositionierung zu finden, erzeugt aber, wie später gezeigt wird, qualitativ unterschiedliche Endpositionierungen.
Der im vorherigen Abschnitt eingeführte Algorithmus STPlaner basiert auf der Tatsache, dass das Finden einer Endpositionierung dem Steinerbaum-Problem entspricht. Damit ist dieses Problem NP-hart und schwer zu lösen. Effiziente Heuristiken lösen das Steinerbaum-Problem in polynomieller Zeit und mit einem kompetitiven Faktor kleiner als 2. So lange sich die Umgebung nicht ändert, bieten solche Heuristiken eine gute Möglichkeit das Navigationsproblem zu lösen. Aufwändig werden diese Verfahren in zwei Fällen:

- bei sehr großen Karten, oder
- bei veränderlichen Umgebungen.

Sollte sich während der Ausführung des Plans ergeben, dass die Umgebung nicht so ist wie erwartet, müssen Verfahren genutzt werden, die die Steinerbäume auf dynamischen Graphen berechnen können wie z.B. in Blin et al. [2009] oder Ding and Ishii [2000] beschrieben. Um diesen Problemen zu begegnen, wird hier der AgentPlaner vorgeschlagen, ein Algorithmus, der nur lokal begrenzte Informationen der Umgebung nutzt. Dieser Algorithmus wird sich am MPR-Flooding-Algorithmus orientieren [Qayyum et al., 2002].

Agenten als Planungswerkzeug

Die Idee, eine Variante des MPR-Flooding für die Suche nach einer gültigen Endpositionierung zu nutzen, ergibt sich aus der Tatsache, dass auch bei der Bestimmung von Routen innerhalb eines Netzwerkes das Steinerbaum-Problem zum Tragen kommt. Der Vorteil liegt hier in der lokalen Suche einer Route. Bei der Bestimmung einer MPR-Menge, wie in Kapitel 3.2.2 beschrieben, werden nur Informationen über die Nachbarschaft benötigt. Dies vereinfacht die Reaktion auf veränderliche Graphen, was insbesondere in der Nachrichtentechnik wichtig ist, da das Netz sich jederzeit ändern kann, indem neue Knoten hinzukommen und andere Knoten wegfallen.
Bei der Entwicklung des Algorithmus AgentPlaner wurde jedoch eine etwas andere Sichtweise als die der Netzwerkkommunikation gewählt. Wie der Name impliziert, wird ein auf Agenten basierender Ansatz gewählt. Dabei werden alle Knoten des Bewegungs- bzw. Bedingungsgraphen als Agenten betrachtet. Jeder Agent kann entweder aktiv oder inaktiv sein. Zusätzlich kann jeder Agent mit allen Agenten kommunizieren, die benachbart sind. Dieser Nachbarschaft kommt eine große Bedeutung zu. Zwei Agenten v_i und v_j sind dann benachbart, wenn v_j in SCG(v_i) mit v_i und v_i in SCG(v_j) mit v_j verbunden

ist. Damit werden die Probleme des STPlaner umgangen und sowohl Informationen des Bewegungs- wie auch des Bedingungsgraphen beachtet. Neben dem Austausch von Informationen untereinander kann ein aktiver Agent seine inaktiven Nachbarn aktivieren.

Expansion des Agentennetzes

Jeder Agent wird durch das Drei-Tupel $\{$aktiv $|$ inaktiv; expandiert $|$ nicht expandiert; Priorität$\}$ und den Knoten, den er repräsentiert, definiert. Zu Beginn des Algorithmus sind alle Agenten abgeschaltet (inaktiv), sie sind nicht expandiert und ihre Priorität ist auf ∞ gesetzt. Ein Agent kann entweder durch die Wahl als Start- oder Zielpunkt oder durch einen anderen aktiven Agenten aktiviert werden.
Durch die Wahl des Startpunktes und der Zielpunkte werden die dazugehörigen Agenten aktiviert und mit höchster Priorität initialisiert. Der aktive Agent bleibt dabei untätig, so lange ein nicht expandierter Agent mit höherer Priorität in der Nachbarschaft vorhanden ist. Existiert kein Agent in der Nachbarschaft, der eine höhere Priorität hat, meldet der Agent an die Nachbarschaft seine Expansion. Dadurch werden andere Agenten an der Expansion gehindert. Die Expansion des Agenten a beinhaltet folgende Schritte:

1. Warte, solange ein benachbarter Agent expandiert.
2. Melde Expansion an benachbarte Agenten.
3. Stelle ein Liste $\mathcal{L}_1$ aller Agenten auf, die in der Umgebung existieren und nicht aktiv sind.
4. Stelle eine Liste $\mathcal{L}_2$ auf, die alle Agenten enthält, die von a zwei Hops entfernt sind und nicht in $\mathcal{L}_1$ liegen.
5. Solange $\mathcal{L}_2$ nicht leer ist:
   (a) Bestimme für jeden Agenten in $\mathcal{L}_1$, wie viele Agenten in $\mathcal{L}_2$ von ihm erreicht werden können.
   (b) Aktiviere den Agenten, der die meisten Agenten in $\mathcal{L}_2$ erreicht, und setze seine Priorität um eins niedriger als die eigene Priorität.
   (c) Entferne die erreichten Agenten aus $\mathcal{L}_2$.
6. Markiere Agent a als *expandiert* und melde die Expansion als abgeschlossen an die Nachbaragenten.

Obwohl der Algorithmus ähnlich zu dem vorgestellten Greedy-Algorithmus zur MPR-Bestimmung ist, existiert eine Besonderheit, die auch der Grund für die Betrachtung als Agentensystem ist. Während bei der MPR-Bestimmung immer nur ein Knoten expandiert wird, können in dem Algorithmus AgentPlaner mehrere Agenten gleichzeitig expandieren. Es handelt sich hierbei um ein asynchron funktionierendes Agentensystem. Dadurch können mehrere Bereiche getrennt voneinander gleichzeitig erfasst werden. Sobald zwei Bereiche sich berühren, sorgen die Agenten durch die Meldung der Prioritäten für eine konsistente Bearbeitung.

Während der Expansion entstehen so zwei Klassen von Agenten, die aktiven und die inaktiven. Während sich die Zahl der aktiven Agenten vergrößert, verringert sich die Zahl der inaktiven Agenten. Um aus den aktiven Agenten eine Lösung für die Endpositionierung zu gewinnen, muss ein Graph, der sogenannte MPR-Graph, extrahiert werden.

Definition 4.17*: MPR-Graph*
Ein MPR-Graph $G_{MPR} = (V, E)$ entsteht aus der Expansion der Agenten. Dabei ist V die Menge aller aktiven Agenten. Zwischen v_i und v_j existiert genau dann eine Kante e, wenn v_j in SCG(v_i) mit v_i und v_i in SCG(v_j) mit v_j verbunden ist.

Damit ist der MPR-Graph (V_g, E_g) der Graph, der alle Knoten enthält, die von den aktiven Agenten repräsentiert werden. Eine Kante zwischen zwei Knoten existiert genau dann, wenn die zugehörigen Agenten benachbart sind.

Abbruchbedingung

Wie oben beschrieben nimmt die Zahl der aktivierten Agenten zu, während die inaktiven Agenten immer weniger werden. Nun ist es für die Suche nach der Endpositionierung nicht nötig, dass alle Agenten aktiviert werden. Die Abbruchbedingung ist dazu da, die Expansion der Agenten zu unterbrechen, wenn das Netzwerk aus aktiven Agenten schon eine Lösung enthält. Dadurch wird die Rechenzeit beschränkt und die Optimierung der Lösung vereinfacht.
Die Abbruchbedingung ist genau dann erfüllt, wenn der MPR-Graph eine Lösung für die Endpositionierung enthält und sie enthält, genau dann erstmals eine Lösung, wenn alle Terminale (Start und Zielpunkte) nach einer Expansion das erste Mal in einer Zusammenhangskomponente des MPR-Graphen liegen.

Lemma 4.18 *Ein MPR-Graph, in dem alle Terminale in einer Zusammenhangskomponete liegen, ist eine gültige Endpositionierung.*

Dabei wird implizit angenommen, dass auf allen Knoten im MPR-Graph, also allen aktiven Agenten, ein Roboter positioniert ist. Für eine gültige Positionierung muss nach Definition 4.7 der Teilgraph des Bedingungsgraphs aus den aktiven Agenten eine Zusammenhangskomponente bilden.

Korollar 4.19 *Wenn im MPR-Graph alle Terminale in einer Zusammenhangskomponente liegen, dann besteht der MPR-Graph nur aus einer Zusammenhangskomponente.*

Beweis. Zu Beginn besteht der MPR-Graph aus genau den Knoten, die die Terminale (Start und Ziele) bilden. D.h. der MPR-Graph besteht aus maximal ebenso vielen Zusammenhangskomponenten. Dabei kann die Anzahl der Zusammenhangskomponenten auch kleiner sein, z.B. wenn Terminale schon benachbart sind.
Durch die Expansion werden nun weitere Knoten dem MPR-Graphen hinzugefügt. Dabei kann ein Knoten nur hinzugefügt werden, wenn er zu einem aktiven Knoten be-

nachbart ist. Somit kann im MPR-Graph durch das Hinzufügen von Knoten keine neue Zusammenhangskomponenete entstehen. Wenn nun alle Terminale in einer Zusammenhangskomponenete liegen, sind damit alle anfänglich existierenden Zusammenhangskomponeneten verbunden und der MPR-Graph besteht aus einer einzigen Zusammenhangskomponente.

■

Damit kann das Lemma 4.18 gezeigt werden:

Beweis. Im MPR-Graph existiert genau dann eine Kante zwischen zwei Knoten, wenn diese benachbart sind. Aufgrund der Definition der Nachbarschaft existiert damit auch immer eine Kante im Bedingungsgraphen zwischen diesen Knoten. Da der MPR-Graph nach Korollar 4.19 nur aus einer Zusammenhangskomponenete besteht, ist der MPR-Graph eine gültige Endpositionierung. ■

Verbesserung des Plans

Nachdem die Abbruchbedingung erfüllt wurde, existiert nun ein MPR-Graph, in dem die Knoten, die den Start und die Zielpunkte repräsentieren, in einer Zusammenhangskomponenete liegen. Damit ist der MPR-Graph eine gültige Endpositionierung. Allerdings enthält er im Allgemeinen noch viele Knoten, die nicht für die Lösung notwendig sind. Außerdem ist der MPR-Graph im Allgemeinen kein Baum. Dies resultiert aus der Tatsache, dass jeder Knoten, der benachbart ist, verbunden ist. Um einen Plan aufbauen zu können, muss dieser Graph in eine Baumstruktur überführt werden. Dies kann über eine Berechnung des Minimal-Spanning-Trees des MPR-Graphen geschehen. Daraus ergibt sich dann der MPR-Baum. Dadurch bleibt die Eigenschaft der gültigen Endpositionierung erhalten. Allerdings bleibt auch die Anzahl der Knoten gleich. Durch die Baumstruktur können jetzt alle Knoten entfernt werden, die keine Terminale sind und den Grad 1 haben. Da das Löschen eines Knoten mit Grad 1 nicht das Zerfallen eines Graphen in mehrere Zusammenhangskomponenten auslösen kann, bleibt die Eigenschaft der Endpositionierung bestehen. Durch das Löschen eines Knotens können weitere Knoten entstehen, die die Löschbedingung erfüllen. Dadurch kann die Anzahl an Knoten reduziert werden, ohne dass der MPR-Baum seine Eigenschaft als gültige Endpositionierung verliert.

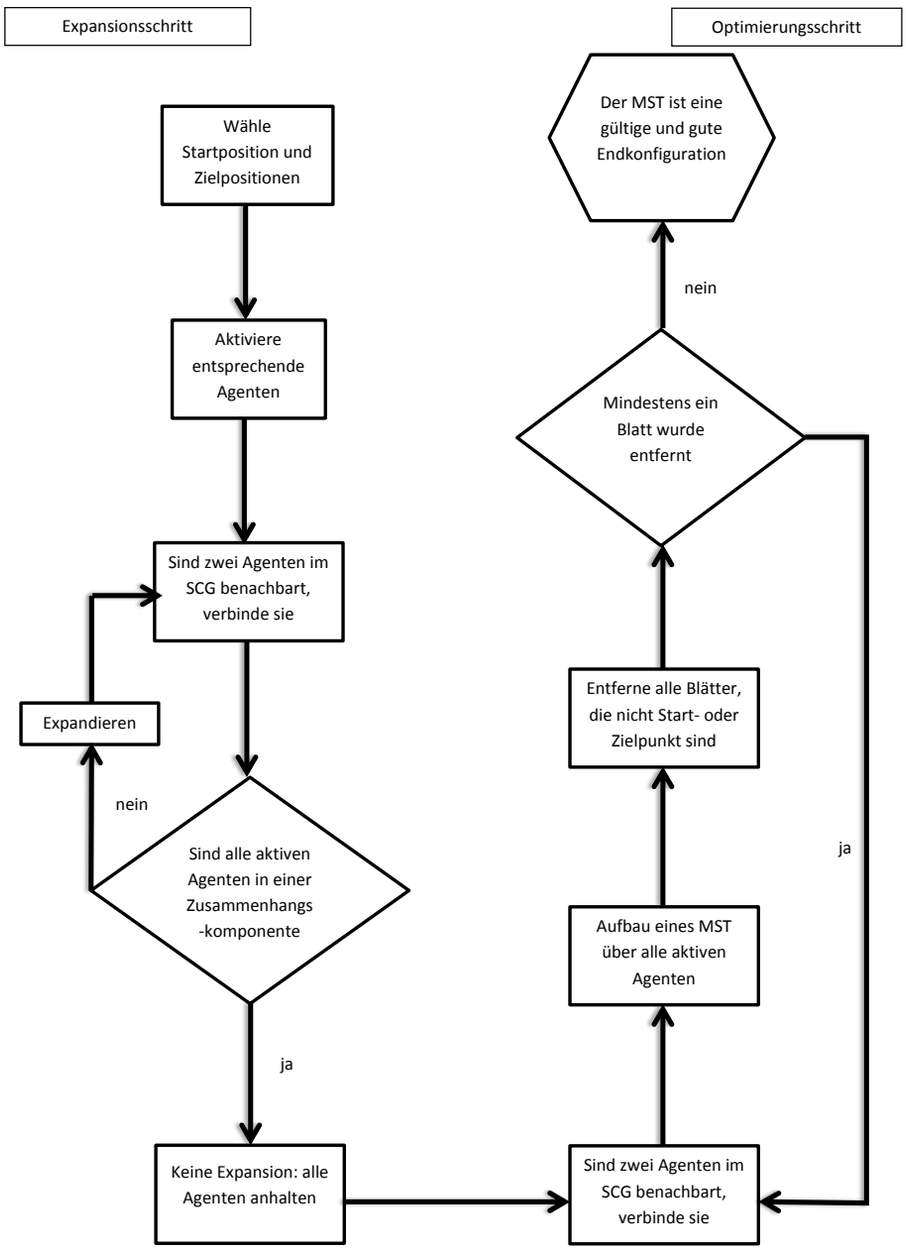


Abbildung 4.10: Ablaufdiagramm des AgentPlaner. Solange der Algorithmus sich in der Expansionsphase befindet, können die Agenten nach den beschriebenen Regeln expandieren. Sobald alle aktiven Agenten eine gemeinsame Zusammenhangkomponente bilden, wird die Optimierungsphase ausgelöst.

Vollständigkeit

Der AgentPlaner berechnet, im Gegensatz zum STPlaner, schon während der Suche nach einer Endpositionierung viele SCGs. Auch wenn diese bei der Erstellung der Pfade zwischen den Zielpunkten später wiederverwendet werden können, so wirkt sich dies doch stark nachteilig auf die Rechengeschwindigkeit aus. Allerdings gewinnt der AgentPlaner hierdurch ohne weiteren Aufwand die Garantie der Vollständigkeit, d.h. wenn eine Lösung existiert, so findet der AgentPlaner auch eine. Existiert keine Lösung, so wird auch dies erkannt.

Lemma 4.20 *Der Algorithmus AgentPlaner ist vollständig, d.h. existiert mindestens eine Lösung, wird auch eine gefunden. Existiert keine Lösung, so wird dies erkannt.*

Zum Beweis der Vollständigkeit müssen folgende Punkte gezeigt werden:

- Existiert eine Lösung, so findet AgentPlaner eine. Das bedeutet für den konkreten Fall, dass der AgentPlaner eine gültige Endpositionierung findet, für die ein gültiger Plan erzeugt werden kann.
- Existiert keine Lösung, so muss der AgentPlaner dies ebenfalls feststellen.

Grundsätzlich können nur Agenten, die in direkter Nachbarschaft zu einem anderen Agenten liegen, aktiviert werden. Das bedeutet, es können maximal alle Agenten aktiviert werden, die in der Zusammenhangskomponente des SCG-Metagraphen liegen. Es ist zu zeigen, dass alle Agenten in dieser Zusammenhangskomponente auch aktiviert werden, wenn keine Abbruchbedingung existiert. Dies zeigt weiterhin auch den Unterschied des AgentPlaner Algorithmus zu einem MPR-Suchalgorithmus aus der Kommunikation.
Damit ein Agent nicht aktiviert wird, darf er im Algorithmus AgentPlaner im Schritt: „Aktiviere den Agenten, der die meisten Agenten in $\mathcal{L}_2$ erreicht, und setze seine Priorität um eins niedriger als die eigene Priorität.“ (Seite 55, Schritt 5(b)) nicht ausgewählt werden. Dies kann durch zwei verschiedene Situationen erreicht werden. Zum einen kann dies passieren, wenn der Agent eine „Sackgasse“ ist, d.h. er erreicht keine weiteren Agenten. Damit hat er im Graph den Grad 1 und kann nur durch den gerade expandierenden Knoten erreicht werden. In Bezug auf die Verschmelzung von zwei Zusammenhangskomponente kann solch ein Agent aber ohnehin keinen Beitrag leisten. Somit ist die Nicht-Aktivierung eines “Sackgassen-Agenten” nicht von Belang für die Lösung der Aufgabe.
Die zweite Möglichkeit, nicht ausgewählt zu werden, besteht darin, solange $\mathcal{A}_2$ nicht leer ist, immer einen Agenten in $\mathcal{A}_1$ zu haben, der mehr Agenten aus $\mathcal{A}_2$ erreicht. Das bedeutet, dass der nicht zu aktivierende Agent keinen Agenten in der unmittelbaren Nachbarschaft hat, den nur er erreichen kann. Damit ist er auch nicht notwendig, um zwei Zusammenhangskomponenten des MPR-Graphen zu verbinden, solange alle seine Nachbarn aktiviert werden.
Mit dieser Vorüberlegung kann Lemma 4.20 bewiesen werden:

Beweis. Wenn der Algorithmus AgentPlaner einen MPR-Graphen findet, so ist dieser eine Endpositionierung. Dies ergibt sich aus dem Beweis 4.18.
Angenommen der Algorithmus findet keine gültige Endpositionierung, die Abbruchbedingung ist also nicht erfüllt. Dies bedeutet, der MPR-Graph muss aus mindestens zwei Zusammenhangskomponenten bestehen. Damit diese zwei Zusammenhangskomponenten nicht vereinigt werden, darf kein MPR-Knoten so eingefügt werden, dass er sowohl Nachbar zu einem Knoten aus der einen, als auch Nachbar zu einem Knoten aus der anderen Zusammenhangskomponente ist. Ein vollständig expandierter MPR-Graph garantiert, nach der vorherigen Argumentation, dass alle Agenten, die zur Verbindung zweier Zusammenhangskomponenten beitragen können, auch aktiviert werden. Damit liegen die Terminale in unterschiedlichen Zusammenhangskomponenten des SCG-Metagraph und es gibt somit keine Lösung. ■

In der Praxis erkennt der AgentPlaner, dass es keine gültige Lösung gibt, sobald keine Agenten mehr expandiert werden können. Damit sind alle Agenten aktiv, sie liegen jedoch nicht in einer Zusammenhangskomponente. Dies kann, je nach Anzahl der Agenten, länger dauern. Da jedoch mit fortschreitender Expansion die Anzahl unerreichter Agenten abnimmt, beschleunigt sich der Expansionsprozess der einzelnen Agenten. So zeigt sich in Versuchen, dass schon am Verhalten der Expansion vom Nutzer erkannt werden kann, dass hier keine Lösung mehr gefunden wird. Dies erkennt man daran, dass durch die Expansion keine neuen Gebiete mehr erschlossen werden, sondern nur noch inaktive Agenten, die von aktiven Agenten umschlossen sind, aktiviert werden.

Zeitbeschleunigte Variante: Der FastAgentPlaner

Wie oben gezeigt, ist vor allem die Kommunikation der Agenten untereinander von entscheidender Bedeutung. Der AgentPlaner nutzt die SCGs, um zu bestimmen, welche Agenten miteinander kommunizieren können, d.h. welche Agenten benachbart sind. Die Berechnung der SCGs ist zeitaufwendig und mit einer anderen Definition der Nachbarschaft kann eine Variante des AgentPlaner erzeugt werden, die bei der Findung von Endpositionierung keine SCGs berechnen muss. Diese Variante, FastAgentPlaner (FAP) genannt, definiert die Nachbarschaft alleine über den Bedingungsgraphen. Im FAP können zwei Agenten kommunizieren, wenn sie im Bedingungsgraph mit einer Kante verbunden sind. Der FastAgentPlaner benötigt nun keine SCGs und in den Simulationen zeigt sich, dass er deutlich schneller eine Endpositionierung findet, als der AgentPlaner. Zudem zeigt sich auch, dass der FastAgentPlaner ebenfalls deutlich schneller als der STPlaner ist (vergleiche Kapitel 5).
Da der FAP jedoch, wie der STPlaner, nur auf dem Bedingungsgraph sucht, verliert man einige Eigenschaften des AgentPlaner. So ist der FastAgentPlaner nicht mehr vollständig, es müssen die gleichen Methoden wie im STPlaner genutzt werden, um die gefundene Lösung auf Ausführbarkeit zu testen. Zudem zeigt sich in den Simulationen, dass auch

die Pläne des FAP nicht einfach auszuführen sind und solche Pläne im Gegensatz zum AgentPlaner wieder temporäre Relaisroboter benötigen.

Güte der Agentenplanung

Im Gegensatz zum STPlaner, bei dem die Güte der Endpositionierung von der gewählten Steinerbaum Heuristik abhängt, ist die Güte des AgentPlaner noch zu untersuchen. Erwartet wird, dass die Leistung des AgentPlaner schon alleine durch die Einschränkung auf eine lokale Suche schlechter sein wird als die des STPlaner.

Bei Betrachtung des Worst-Case Szenarios zeigt sich dann auch, dass eine vom AgentPlaner gefundene Endpositionierung beliebig viel schlechter sein kann als die optimale Lösung. Dies resultiert aus der Tatsache, dass der AgentPlaner neben den Nebenbedingungen auch die möglichen Bewegungen betrachtet. So ist es möglich, eine Umgebung zu schaffen, in der sämtliche Knoten zur Endpositionierung vom AgentPlaner hinzugefügt werden, die optimale Lösung aber lediglich zwei Knoten beinhaltet. Eine solche Umgebung ist z.B. die Umgebung aus Abbildung 4.9. Hierbei besteht die optimale Endpositionierung nur aus S und V, also aus zwei Knoten. Um jedoch einen Roboter von S nach T bewegen zu können, ist es nötig, weitere temporäre Relais-Roboter auf die nummerierten Knoten zu stellen (wie in Lemma 4.9 beschrieben). Endpositionierungen, die durch den AgentPlaner gefunden werden, enthalten jedoch keine TRRs (temporäre Relais-Roboter), da es immer einen direkten Weg zwischen zwei PRRs (permanente Relais-Roboter) in der Endpositionierung des AgentPlaner gibt. Da in Abbildung 4.9 der einzig mögliche Weg jedoch über die nummerierten Knoten führt, wird jeder dieser nummerierten Knoten in der Endpositionierung des AgentPlaner auftauchen. Da dieser Weg beliebig lang gestaltet werden kann, enthält die Endpositionierung des AgentPlaner beliebig mehr PRRs als die Lösung des STPlaner. Hier ist jedoch zu beachten, dass dies nicht bedeutet, dass die Lösung des AgentPlaner mehr Roboter benötigt als die Lösung des STPlaner. Zur Ausführung müssen beim STPlaner auch die entsprechenden TRRs bereitgestellt werden.

Bei der Betrachtung der Güte des AgentPlaner ist allerdings zu beachten, dass hier schon bei der Berechnung der Endpositionierung die Bewegung der Roboter zu den Positionen hin berücksichtigt wird. So ist zwar die Anzahl der PRRs in der Endpositionierung beliebig schlechter im Vergleich zur optimalen Lösung, bei der späteren Ausführung des Planes sind solche Endpositionierungen aber meist besser als die des STPlaner.

Eine genaue Evaluation des AgentPlaner und des FastAgentPlaner wird in Kapitel 5 durchgeführt. Dabei werden die Leistungen der Agenten-basierten Verfahren mit denen des STPlaner verglichen.

4.3 Wegfindung unter Nebenbedingungen

Wie oben gezeigt, kann eine Endpositionierung auf verschiedene Weisen gefunden werden. Mit dem STPlaner, dem AgentPlaner und dem FastAgentPlaner sind in dieser Arbeit drei Algorithmen entwickelt worden, die Endpositionierungen finden, die auch der Nebenbedingung genügen. Bei der Problemdefinition der koordinierten Navigation unter Nebenbedingungen wurde zudem gefordert, dass die Nebenbedingung auch erfüllt, ist während die Roboter zur Endpositionierung fahren. Daher stellt sich das Problem, aus der Endpositionierung einen sogenannten globalen Mehrroboterplan zu erstellen, der neben den Zielpunkten auch die Pfade angibt, auf denen die Roboter fahren dürfen. Hierfür werden wieder die SCGs genutzt.
Für die weitere Betrachtung innerhalb dieses Unterkapitels müssen die Pfade, bezogen auf das Problem der koordinierten Navigation mit spatialen Nebenbedingungen, genauer betrachtet werden. Ein gültiger Pfad Π_a^b von **a** nach **b** darf nicht die Nebenbedingung verletzten. Um solch einen Pfad zu beschreiben, werden die SCGs genutzt. Ist **b** in SCG(**a**) mit **a** verbunden, so existiert nach Definition der SCGs ein direkter Weg von **a** nach **b**. Dieser direkte Weg kann einfach im zugehörigen SCG(**a**) gefunden werden. Daher reicht es, SCG(**a**) als Pfad anzugeben. Liegt **b** nicht in SCG(**a**) oder ist nicht verbunden, so muss eine Kette von SCGs von **a** nach **b** gefunden werden. Ein Pfad von **a** nach **b** könnte folgendermaßen aussehen: SCG(**a**); SCG(v_1); SCG(v_2); SCG(v_3); SCG(v_4). Dabei muss v_1 in SCG(**a**) liegen und mit **a** verbunden sein. Ebenso muss v_2 in SCG(v_1) mit v_1 verbunden sein, und so weiter. Zuletzt muss dann **b** in SCG(v_4) mit v_4 verbunden sein.

Algorithm 1 : Pfadplanung

1: $x \leftarrow a$
2: $postmp \leftarrow$leer
3: $path \leftarrow$leer
4: **while** $x \neq b$ **do**
5: Finde ein y, welches in SCG(x) mit x verbunden ist und für das gilt: $d(y, b) < d(x, b)$
6: $path \leftarrow path$ + Pfad von x nach y in G_{Bew}^{sub}
7: $x \leftarrow y$
8: **end while**

Algorithmus 1 beschreibt das allgemeine Pfadplanungsproblem. Die Schwierigkeit liegt dabei in der Zeile 5 des Pseudocodes. Es ist nicht einfach, wenn überhaupt möglich, eine Metrik zu definieren, die angibt, ob zwei Knoten auf den Pfad bezogen näher aneinander liegen als andere. So ist z.B. die euklidische Distanz, wie im Beispiel der Abbildung 4.9 zu sehen, nicht geeignet. Es kann sein, dass ein SCG(v_i), welches die Lösung für Π_a^b ist, im euklidischen Raum weiter von **b** entfernt ist als **a**. Dennoch ist es der kürzeste Weg nach der Pfaddefinition innerhalb dieses Problemkreises.

Für eine beliebige Nebenbedingung ist derzeit keine nicht-triviale, zulässige Heuristik für eine A*-Suche bekannt. Daher wird die Nutzung der Breitensuche, die der A*-Suche mit trivialer Heuristik entspricht, vorgeschlagen. Diese Breitensuche hat eine Komplexität von maximal $O(n)$ bei n Knoten und zusätzlich eine Optimalität in Bezug auf die Anzahl der SCGs. Die Pfadsuche wird implementiert durch Algorithmus 2.

Algorithm 2 SCG-Pfadsuche (SCGPfad)

```
1:  procedure SCGPFAD(Start a, Ende b)
2:      searchtree.wurzel ← a
3:      füge alle Knoten zu searchtree als Blätter von a hinzu, die im SCG(a) mit a
    verbunden sind
4:      Markiere diese Blätter als unbesucht
5:      while b ∉ searchtree und mindestens ein Blatt in searchtree ist unbesucht do
6:          EXPAND-LEAF
7:      end while
8:      if b ∈ searchtree then
9:          RETURN Pfad von a nach b in searchtree
10:     else
11:         RETURN Kein Weg vorhanden
12:     end if
13: end procedure
14:
15: function EXPAND-LEAF
16:     x ← erstes nicht besuchtes Blatt mit geringster Tiefe in searchtree
17:     markiere x als besucht
18:     füge alle Knoten, die noch nicht in searchtree sind und in SCG(x) mit x ver-
    bunden sind, als Blätter zu x hinzu
19:     Markiere diese Blätter als nicht besucht
20: end function
```

Der Pfad, der von diesem Algorithmus ausgegeben wird, enthält alle Positionen, auf die die temporären Relais-Roboter für den Weg von **a** nach **b** positioniert werden müssen. Zusätzlich ist der Weg zwischen diesen temporären Robotern befahrbar, ohne die Nebenbedingung zu verletzen.
Der gesamte Weg kann folgendermaßen erzeugt werden: Seien x_0 bis x_n die Positionen für die zusätzlichen temporären Relais-Roboter. Also muss zuerst ein Roboter von **a** nach x_0 bewegt werden. Dazu wird der Weg im SCG(**a**) gesucht. Als nächste Teilstrecke wird entsprechend der Weg von x_0 nach x_1 im SCG(x_0) gesucht. Dies wird für alle Teilwege bis **b** durchgeführt. Zusammengesetzt ergibt sich der gesamte Pfad. Dies wird in Algorithmus 3 dargestellt. Die Korrektheit und Vollständigkeit des Algorithmus wird in Anhang B.2 bewiesen.

Solch ein Pfad muss für jedes Vorgänger-Nachfolger-Paar aus dem Steinerbaum extrahiert werden. So kann für jede Vorgänger-Nachfolger Beziehung im Steinerbaum solch ein Pfad berechnet werden (siehe Algorithmus 2). Werden die Pfade dann beginnend von der Wurzel (dem Startpunkt) ausgeführt, muss beachtet werden, dass ein Roboter erst dann in solch ein Vorgänger-Nachfolger-Paar eintreten darf, wenn der Vorgänger von einem Roboter besetzt ist. So entsteht ein Plan, d.h. eine Ausführungsanweisung mit einer zeitlichen Komponente.

Algorithm 3 Pfad Extraktion

```
1:  procedure EXTRACT-PATH(Pfad P)
2:      index ← 0
3:      Pfad out ← leer;
4:      while index + 1 ≠ P.size() do
5:          i ← P[index]
6:          j ← P[index + 1]
7:          out ←SCG(i).getPath(j)
8:      end while
9:      RETURN out
10: end procedure
```

Algorithm 4 STPlan

```
1:  procedure STPLAN(Start s, Ziele z, Umgebung E, Bedingung C)
2:      Graph G_Bew ← Bewegungsgraph(E)
3:      Graph G_Bed ← Bedingungsgraph(E, C)
4:      Terminale ← Start + Ziele
5:      Graph Endpositionierung ← Steinerbaum(Terminale, G_Bed)
6:      for jedes Paar Elter-Kind (a,b) aus Endpositionierung do
7:          Pfad P ← SCGPfad(a,b)
8:          Setzte für Paar (a,b) ← EXTRACT-PATH(P)
9:      end for
10: end procedure
```

4.4 Simulationsexperimente

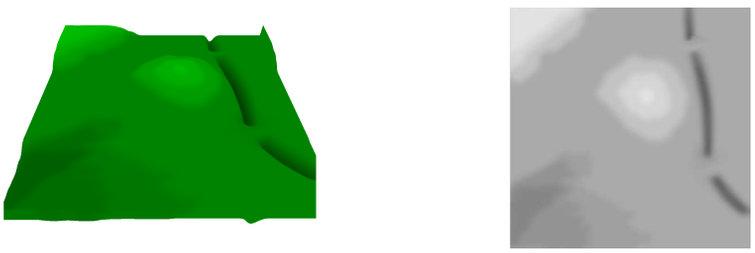
In diesem Abschnitt werden Beispiele und Ergebnisse der Planungsalgorithmen STPlaner, AgentPlaner und FastAgentPlaner präsentiert. Dabei werden typische Situationen besprochen und die Vor- und Nachteile der Verfahren gezeigt.

4.4.1 Simulierte Welten

Um die Eigenschaften der Pläne zu demonstrieren, werden zwei verschiedene simulierte Umgebungen geschaffen:

- Landscape (Abbildung 4.11): eine hügelige Landschaft. Auffällig sind hier vor allem der Hügel im Zentrum und der Graben im rechten Bereich. So kann zwischen den Bereichen rechts des Grabens und links des Grabens nur mittels zwei Brücken gewechselt werden.
- Canyon: (Abbildung 4.12): eine Canyon-ähnliche Landschaft. Ein stilisierter Canyon mit jeweils einem Aufgang zu den Flanken, wobei die rechte Flanke niedriger liegt als die linke Flanke.

Beide Umgebungen sind generiert worden, um die Reaktion der Planungsalgorithmen auf verschiedenen Geländegegebenheiten zu testen. Die Landscape-Umgebung bietet zunächst eine große, nur leicht gewellte Fläche. Solch eine Geländeformation sollte für die Planung einfach sein und auch die Nebenbedingungen nicht beeinflussen. Die beiden Berge bilden dabei Hindernisse, da ihre Flanken recht steil sind und nur an bestimmten Punkte überwunden werden können (siehe Bewegungsgraph im nächsten Abschnitt). Unterbrochen wird die gewellte Fläche durch einen scharfen Einschnitt, einen Graben. Dieser kann nur mit Hilfe zweier Brücken überwunden werden. Diese Engstellen müssen vom Planungsalgorithmus beachtet werden, insbesondere, da die Nebenbedingungen nicht von dem Graben beeinflusst werden.

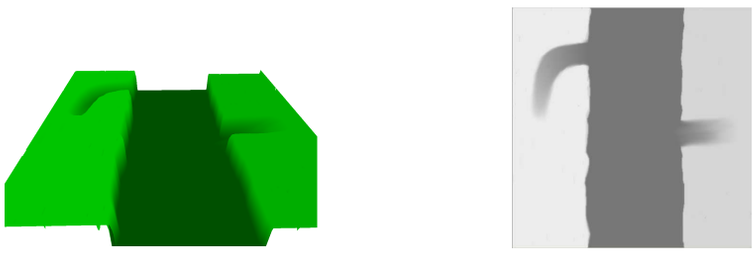


(a) (b)

Abbildung 4.11: Beispielumgebung 1: Landscape. (a) 3D-Darstellung (b) 2D-Schema. Die Höhe ist über den Grauwert kodiert. Dadurch sind niedrige Gebiete dunkler, höhere Gebiete heller.

Die Canyon Umgebung stellt in vielen Punkten ein Gegenteil der Landscape-Umgebung dar. Hier sind viele scharfe Brüche enthalten. Insbesondere fällt die Unterteilung in zwei sich gegenüber liegenden Plateaus sowie den dazwischen liegenden Graben auf. Die scharfe Abbruchkante zum Graben hat einen großen Einfluss sowohl auf die Nebenbedingungen wie auch auf die Bewegungen der Roboter. Die Rampen haben hohe Wände, sodass insbesondere die runde Rampe nur sehr schlecht einzusehen ist. Insgesamt wird

erwartet, dass das Planen und Koordinieren auf der Canyon-Umgebung schwieriger ist, als in der Landscape-Umgebung.



(a) (b)

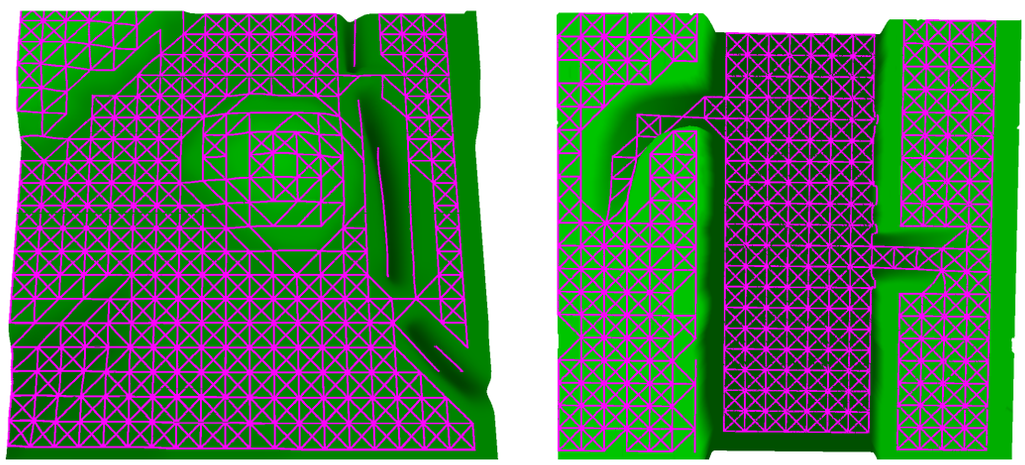
Abbildung 4.12: Beispielumgebung 2: Canyon. (a) 3D-Darstellung (b) 2D-Schema

4.4.2 Bewegungsgraphen in den simulierten Umgebungen

Der Bewegungsgraph ist ein Roadmap-Graph mit gleichförmig verteilten Knoten in der Umgebung. Zwei benachbarte Knoten sind genau dann miteinander verbunden, wenn der Roboter von einem zum anderen gelangen kann. In der hügeligen Landschaft (siehe Abbildung 4.13a) ist ein Großteil der Fläche frei befahrbar. Der Graben im rechten Bereich der Umgebung trennt dabei den Bewegungsgraphen auf, sodass der rechte obere Bereich nur über zwei Brücken zu erreichen ist. Die Hügel (in der Mitte und links oben) sind nur auf bestimmten Wegen zu befahren, da sonst die Steigung für den bodengebundenen Roboter zu stark ist. In den Gräben sind ebenfalls Wege des Bewegungsgraphen zu erkennen. Dies bedeutet, ein Roboter im Graben kann darin herumfahren. Jedoch sind diese Teile nicht mit dem Rest des Bewegungsgraphen verbunden, so dass kein Roboter in den Graben hinein gelangen kann.
Der Bewegungsgraph in der Canyon-Umgebung (in Abbildung 4.13b) besteht im Wesentlichen aus drei Flächen. Jede dieser Flächen ist nahezu vollverknüpft. Die Flächen sind untereinander nur über schmale Rampen verbunden, sodass die Herausforderung hier hauptsächlich im Wechsel von einem Bereich in einen anderen Bereich liegt. Gerade die scharfen Abbruchkanten des Canyon kennzeichnen diese Umgebung.

4.4.3 Bedingungsgraphen in den simulierten Welten

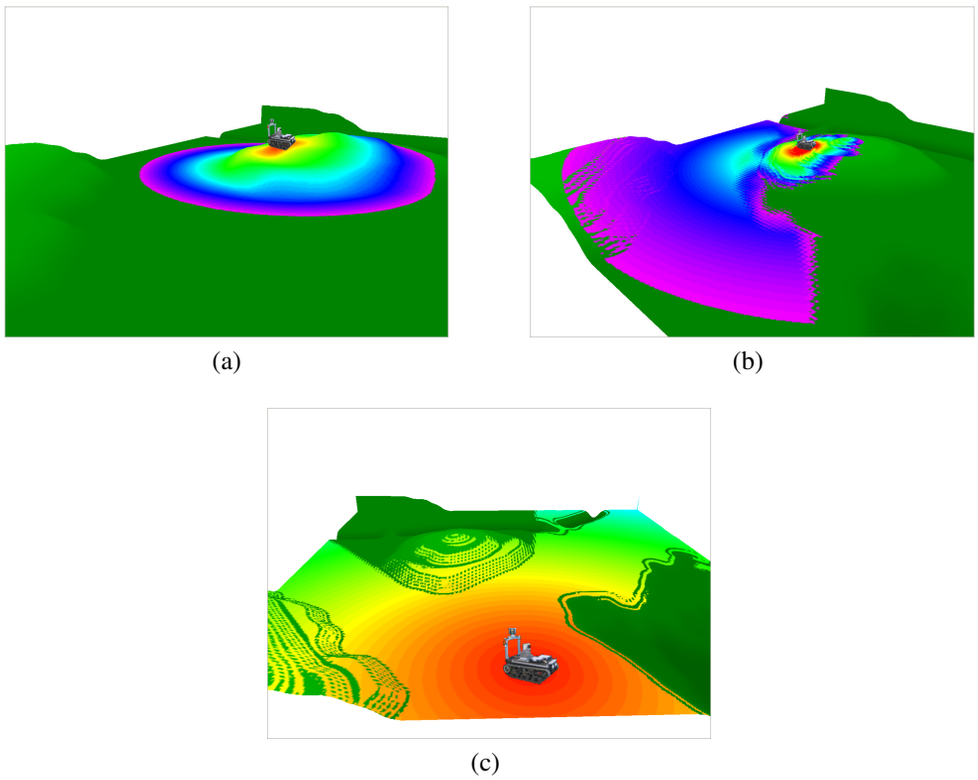
Die Bedingungsgraphen sind sowohl für jede der beiden simulierten Umgebungen wie auch für jede Nebenbedingung sehr unterschiedlich. Zudem enthält der Bedingungsgraph eine sehr große Menge an Kanten, sodass er nicht sinnvoll vollständig visualisiert werden kann. Daher wird nur jeweils ein Bereich um einen bestimmten Knoten beispielhaft gezeigt. Im Weiteren werden folgende Nebenbedingungen betrachtet:



(a) (b)

Abbildung 4.13: Bewegungsgraphen in der Simulation. (a) Bewegungsgraph für einen bodengebundenen Radroboter. Dabei wird eine bestimmte Steigung angenommen, die der Roboter maximal bewältigen kann. So sind einige Flanken der Hügel nicht befahrbar. Ebenso kann der Graben nur über die Brücken passiert werden. Die Teilgraphen in den Gräben bedeuten, dass, wenn der Roboter im Graben starten würde, er auch am Boden desselben fahren könnte. In diesem Bewegungsgraphen sind auch Diagonalen erlaubt. (b) Bewegungsgraph für den Canyon für einen bodengebundenen Radroboter. Auch hier ist die Steigung der limitierende Faktor. Die beiden seitlichen Plateaus können nur über die Rampen erreicht werden. In diesem Bewegungsgraphen sind auch Diagonalen erlaubt.

- Distanznebenbedingung:
  Die Distanznebenbedingung fordert für die Roboter, dass jeder Roboter nicht zu weit von seinem Vorgänger entfernt sein darf. Damit sind alle Knoten, die näher als eine bestimmte Distanz vom betrachteten Knoten entfernt sind, im Bedingungsgraph verbunden. Es zeigt sich ein Kreis um den Knoten, in dem die Nebenbedingung erfüllt ist. Dabei wird die Nebenbedingung nicht von der Umgebung beeinflusst (siehe Abbildung 4.14a).
- Kommunikationsnebenbedingung:
  Die Kommunikationsnebenbedingung bildet die Kommunikationsfähigkeit der Roboter ab. Mit ihr soll sichergestellt werden, dass alle Roboter miteinander kommunizieren können. Um vorherzusagen, ob zwischen zwei Punkten eine Kommunikation möglich ist, kann ein Wellenausbreitungsmodell genutzt werden. In dem einfachen hier benutzten Modell von Goldhirsh and Vogel [1998] wird angenommen, dass sich das Signal über die Distanz logarithmisch verschlechtert. Zusätzlich wird es an Hindernissen linear gedämpft (siehe Abbildung 4.14b).
- Sichtbarkeitsnebenbedingung:



(a) (b)

(c)

Abbildung 4.14: Die Nebenbedingung wird für den eingezeichneten Roboter betrachtet. Jeder Punkt innerhalb der markierten Umgebung erfüllt die Nebenbedingung. (a): Distanznebenbedingung (b): Kommunikationsnebenbedingung (c) Sichtbarkeitsnebenbedingung

Die Sichtbarkeitsnebenbedingung verlangt, dass jeder Roboter von seinem Vorgänger zu jeder Zeit gesehen werden kann. Dies ist vor allem in gefährlichen Umgebungen von Vorteil. So kann bei einem eventuellen Ausfall eines Roboters visuell aufgeklärt werden, wodurch dieser Ausfall ausgelöst wurde. Diese Nebenbedingung ist natürlich stark von der Umgebung abhängig (siehe Abbildung 4.14c). Dabei sind auch Bereiche, die tiefer liegen, von der relativ niedrigen Position der Kamera des Roboters nicht einzusehen. Im Allgemeinen wird bei dieser Nebenbedingung angenommen, dass die Sichtweite unbeschränkt ist. Durch eine Kombination der Sichtbarkeitsbedingung mit der Distanzbedingung kann eine Einschränkung des Sichtbereichs aber ebenfalls berücksichtigt werden.

4.4.4 Beispielplanungen

Die drei vorgestellten Algorithmen STPlaner, AgentPlaner und FastAgentPlaner berechnen die Endpositionierung unterschiedlich. Daher wird erwartet, dass sie in den

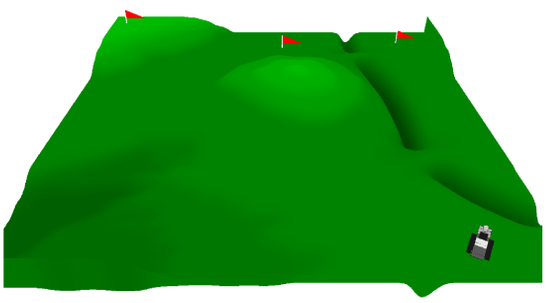
unterschiedlichen Umgebungen unterschiedliche Eigenschaften zeigen. Um diese Eigenschaften zu charakterisieren und Unterschiede zwischen den Planern darzustellen, werden in diesem Unterkapitel beispielhaft verschiedene Planungen durchgeführt. Dazu werden alle drei vorgestellten Algorithmen sowohl auf der Landscape-Umgebung als auch auf der Canyon-Umgebung eingesetzt. So ergibt sich ein erster Eindruck von den Algorithmen, der in Kapitel 5 auch quantitativ bestätigt wird.

STPlaner

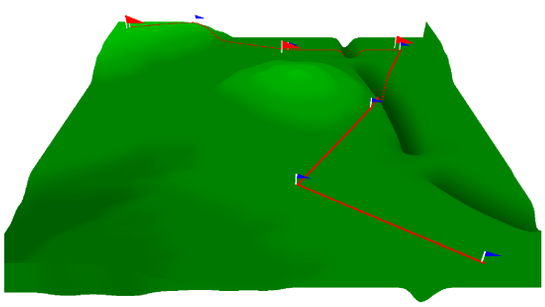
Der STPlaner nutzt die Steinerbaum-Heuristik nach Melhorn zur Erstellung der Endkonfiguration. Dadurch enthalten die Endkonfigurationen nur wenige Knoten und benötigen wenige PRR. Bei Betrachtung der daraus entstehenden Wege zur Endkonfiguration zeichnet sich als ein Nachteil des STPlaner ab, dass durch die Minimierung der PRRs viele TRRs benötigt werden. Dies schlägt sich in sehr komplexen und langen Wegen nieder.
In der hügligen Landschaft erweist sich dies als unproblematisch, vergleiche Abbildung 4.15. Die berechnete Endpositionierung kann über einen recht einfachen Weg erreicht werden. Anders sieht dies für die Canyon-Umgebung in Abbildung 4.16 aus. Auch hier degeneriert der Steinerbaum zu einem Linienzug. Es ist aber schon zu sehen, dass die Reihenfolge, in der die Steinerknoten angefahren werden müssen, für die Ausführung ungeschickt ist. So wird der Weg immer zwischen den einzelnen Plateaus hin und her wechseln. Dies erklärt den kompliziert verlaufenden Pfad. Zusätzlich – und nicht in den Abbildungen zu sehen – werden viele temporäre Relais-Roboter benötigt, um diesen komplexen Plan auszuführen, ohne die Nebenbedingung zu verletzten. In diesem Fall werden 6 zusätzliche TRR benötigt.

AgentPlaner

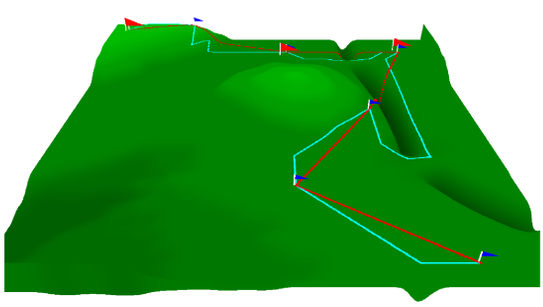
Der AgentPlaner beachtet, im Gegensatz zum STPlaner, bei der Planung des globalen Planes zusätzlich die Bewegungsmöglichkeiten der Roboter. Dadurch wird zwar eine Endpositionierung erwartet, die mehr Roboter benötigt als die des STPlaner; ein solch komplexer Pfad wie in Abbildung 4.16 wird in aller Regel aber vermieden. In der Planung für das hüglige Gelände ist zwischen den Resultaten des STPlaner und des AgentPlaner kaum ein qualitativer Unterschied auszumachen (vergl. Abbildung 4.17). In der Canyon-Umgebung zeigt sich die erwartete Verhaltensweise. Die resultierende Endkonfiguration in Abbildung 4.18e beinhaltet gegenüber der Endkonfiguration des STPlaners vier weitere Roboter. Der resultierende Pfad (Abbildung 4.18f) ist deutlich einfacher. Hier ist anzunehmen, dass solch ein Pfad auch realistisch auszuführen ist.



Initiale Situation für den Planer. Der Roboter markiert den Startpunkt, die drei roten Fahnen die Zielpunkte.

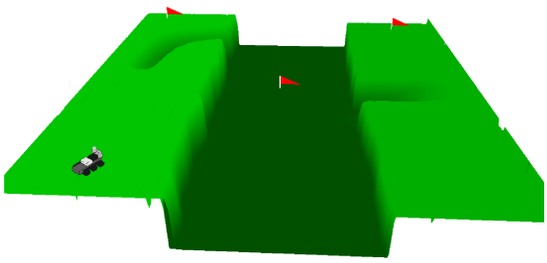


Der errechnete Steinerbaum (rot) degeneriert hier zu einem Linienzug.

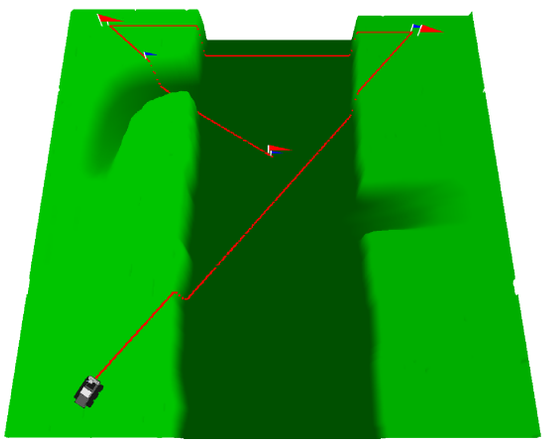


Der Pfad folgt dem Steinerbaum, muss aber den Umweg über die Brücken nehmen.

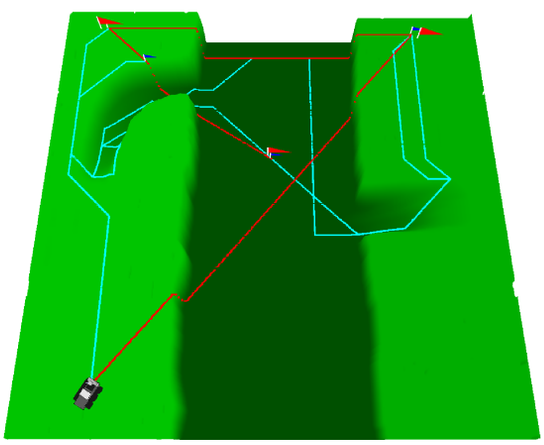
Abbildung 4.15: Beispielplanung des STPlaners in der Landscape Umgebung: Planung auf der hügligen Umgebung mit der Kommunikations-Nebenbedingung



Initialer Zustand der Planung. Der Roboter markiert die Startposition, die roten Fahnen die Endkonfigurationen.

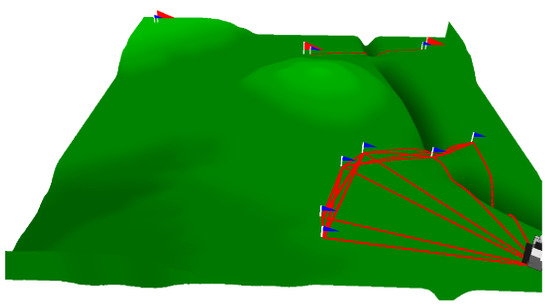


Der berechnete Steinerbaum (rot).

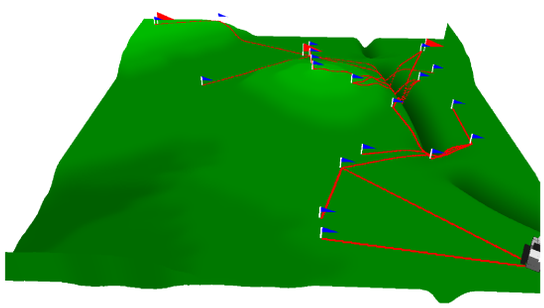


Der aus der Endkonfiguration berechnete Weg ist sehr komplex und umständlich. Dies resultiert daraus, dass die Reihenfolge der anzufahrenden Positionen ungeschickt gewählt ist.

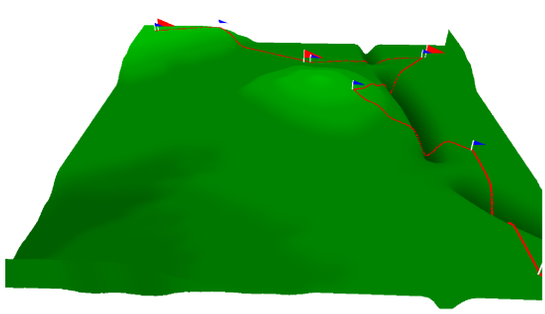
Abbildung 4.16: Beispielplanung des STPlaners in der Canyon Umgebung: Planung im Canyon mit der Sichtbarkeits-Nebenbedingung.



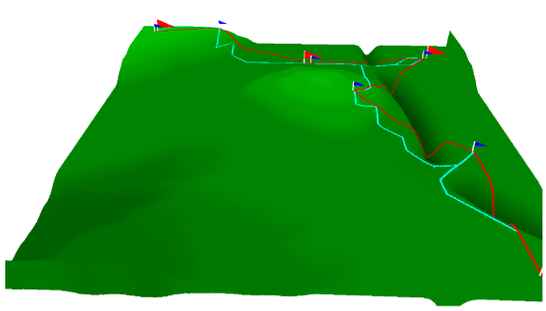
Initialer Zustand des Planers: der Startpunkt wird durch den Roboter markiert, die Zielpunkte durch die roten Fahnen. Der Startpunkt ist zu diesem Zeitpunkt schon expandiert.



Die Expansion ist abgeschlossen und der Startpunkt sowie alle Zielpunkte liegen in einer Zusammenhangskomponente. Noch sind viele Knoten enthalten, die nicht für die Endkonfiguration notwendig sind.

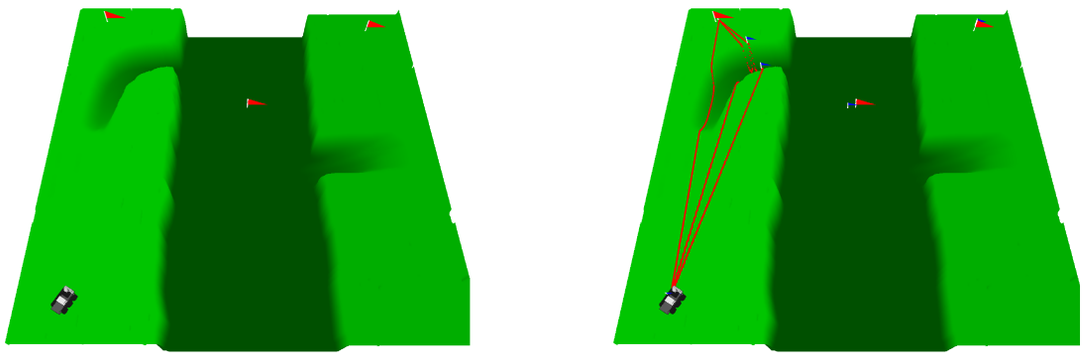


Die berechnete Endkonfiguration nach Beendigung des Optimierungsschrittes. Qualitativ unterscheidet sie sich kaum von der Endkonfiguration des STPlaner.



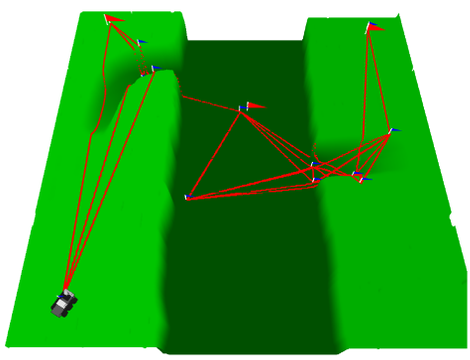
Der berechnete Weg (türkis) teilt sich oben in der Mitte, um den Zielpunkt am rechten Rand sowie die beiden anderen Zielpunkte zu erreichen.

Abbildung 4.17: Beispielplanung des AgentPlaner in der Landscape-Umgebung: Nebenbedingung ist die Aufrechterhaltung der Kommunikation.

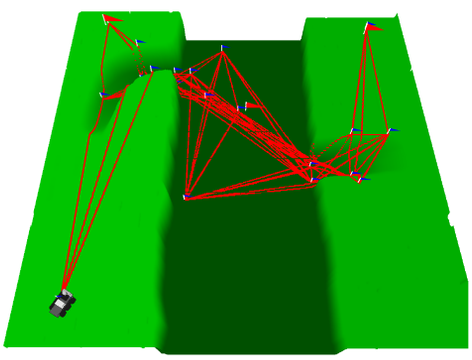


(a)

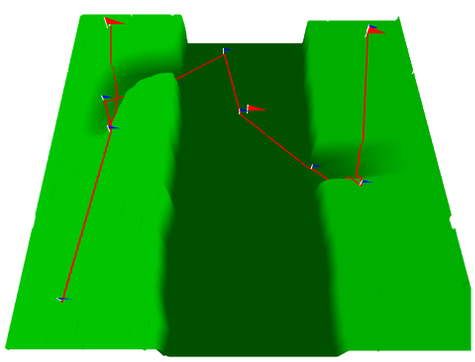
(b)



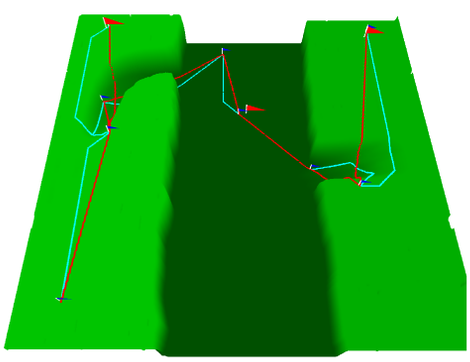
(c)



(d)



(e)

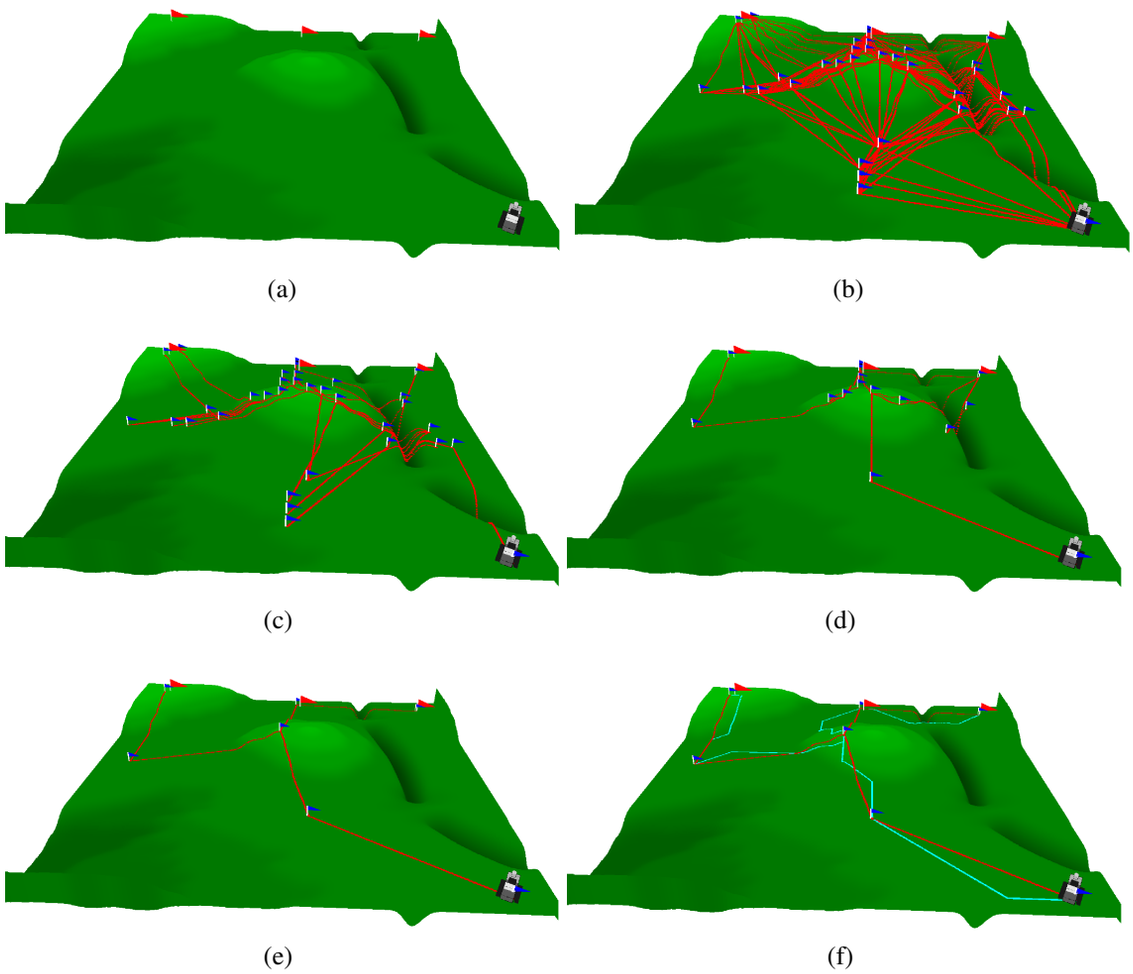


(f)

Abbildung 4.18: Beispielplanung des AgentPlaner in der Canyon-Umgebung mit der Nebenbedingung Sichtbarkeit. a) Initaler Zustand des Planers; b) erste Expansion des Startknotens; c) die Expansion ist weit fortgeschritten, jedoch sind der linke und der rechte Teil noch nicht verbunden; d) die Expansion ist vollständig, alle Zielpunkte sind mit dem Startpunkt in einer Zusammenhangskomponente; e) durch den Optimierungsschritt gewonnene Endkonfiguration; f) der resultierende Pfad.

FastAgentPlaner

Der FastAgentPlaner(FAP) ist eine Variante des AgentPlaner. Ebenso wie dieser benutzt der FAP lokale Informationen, um eine Suche auf den möglichen Positionierungen durchzuführen. Da sowohl beginnend vom Startpunkt wie auch von den Zielpunkten gesucht wird, beschleunigt sich die Suche. Der FastAgentPlaner nutzt dabei nicht wie der AgentPlaner sowohl den Bedingungsgraph als auch den Bewegungsgraph bei der Suche aus. Der FAP sucht nur auf dem Bedingungsgraph und muss daher während der Suche keine SCGs berechnen. Dies schlägt sich in einer deutlich erhöhten Geschwindigkeit während der Suche nach einer Endkonfiguration nieder. Es bedingt zugleich auch die Nachteile, die aus dem STPlaner bekannt sind. Somit erbt der FAP vom AgentPlaner die etwas schlechteren Endkonfigurationen (verglichen mit dem STPlaner) und vom STPlaner die schwerer auszuführenden Pläne. Dies ermöglicht eine deutlich höherer Berechnungsgeschwindigkeit (für genauere Werte siehe Kapitel 5). So zeigt die Beispielplanung in Abbildung 4.19, dass die resultierende Endkonfiguration einen Roboter mehr benötigt als die Beispielplanung des STPlaner. Innerhalb der Canyon-Umgebung (siehe Abbildung 4.20) entsteht auch ein komplexer Pfad, ähnlich zum STPlaner (Abbildung 4.20f).

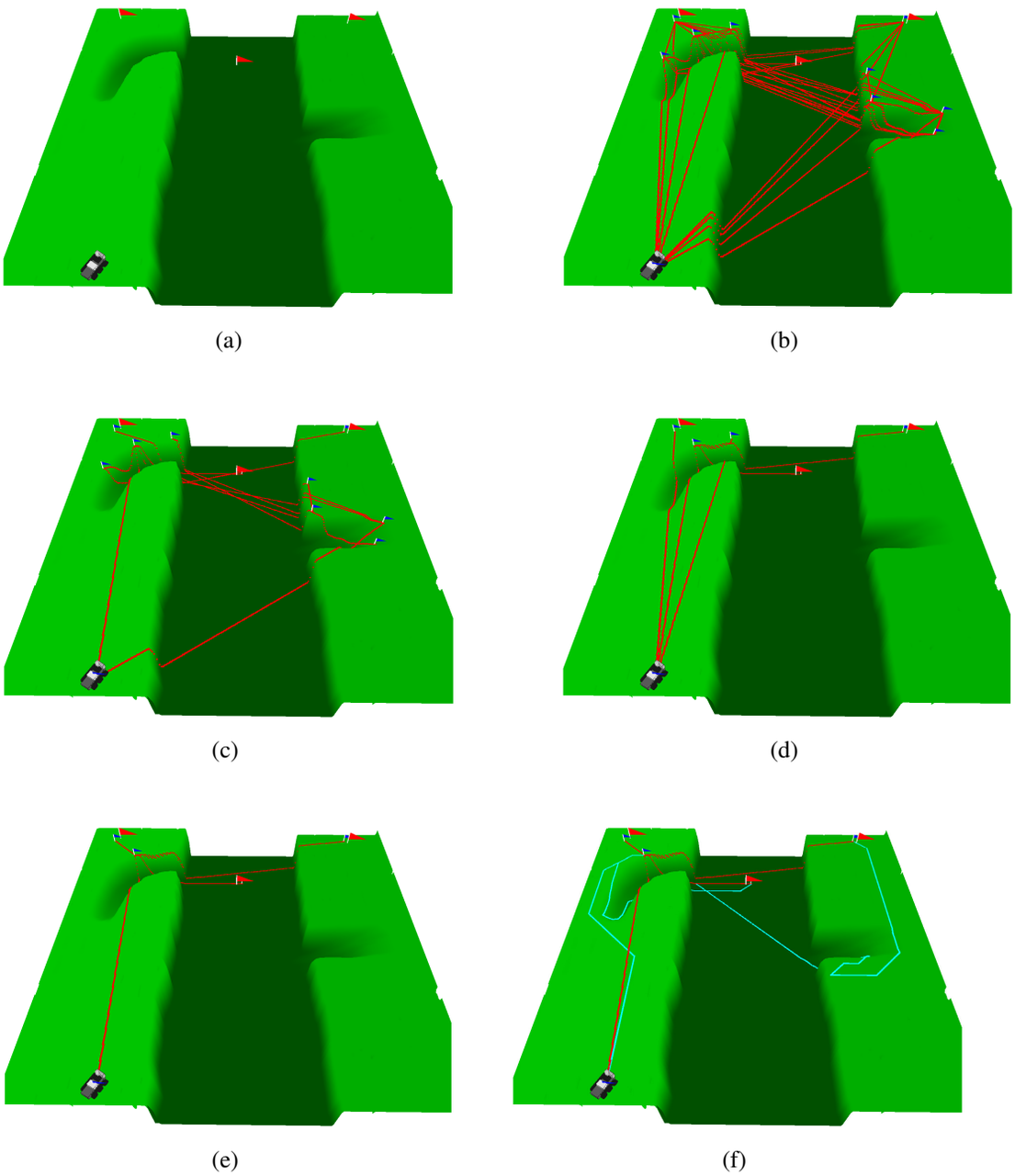


(a) (b)

(c) (d)

(e) (f)

Abbildung 4.19: Beispielplanung des FastAgentPlaner in der Landscape-Umgebung mit der Nebenbedingung Kommunikation. a) Initiale Situation des Planers. Der Roboter repräsentiert den Startpunkt, die roten Fähnchen die Zielpunkte. b) Die Suchstruktur des Planers zum Zeitpunkt der Erfüllung der Abbruchbedingung; c) erster Lösungsbaum, hier noch mit vielen nicht notwendigen Knoten; d) ein Lösungsbaum während des Optimierungsschrittes; e) resultierender Steinerbaum; f) resultierender Pfad (türkis).



(a) (b)

(c) (d)

(e) (f)

Abbildung 4.20: Beispielplanung des FastAgentPlaner in der Canyon-Umgebung mit der Nebenbedingung Sichtbarkeit. a) Initialer Status des Planers. b) Vollständiger MPR-Graph des FAP. c) MPR-Graph nach Reduzierung auf einen Baum. d) MPR-Graph während des Optimierungsschrittes. e) Resultierender Steinerbaum. f) Resultierender Pfad (türkis).

4.5 Von globalen Mehrroboterplänen zu lokalen Handlungsanweisungen

Um den Schritt von der Planung zu den realen Robotern zu gestalten, muss der gefundene Mehrroboterplan in Roboterbewegungen, hier Handlungsanweisungen genannt, umgesetzt werden. Dabei werden die globalen Mehrroboterpläne in Anweisungen mit einer räumlichen und einer zeitlichen Komponente umgesetzt. Die Roboter müssen dazu wissen, wann sie zu welchen Zielpunkt fahren dürfen. Insbesondere der zeitliche Aspekt garantiert dabei, dass die Nebenbedingung nicht verletzt wird.
Der Schwerpunkt dieser Arbeit liegt in der Betrachtung der globalen Planung unter Nebenbedingungen, daher wird an dieser Stelle nicht detailliert auf die Erstellung der Handlungsanweisungen eingegangen. Dennoch war auch dieser Themenkomplex zu bearbeiten, da andernfalls keine Experimente auf realen Systemen vorgenommen werden konnten. Dabei stellte sich zunächst die Frage, wie viele Roboter zur Erfüllung eines globalen Mehrroboterplanes nötig sind. Dazu wurden die sogenannten „Navigation Trees“ eingeführt. Diese sind eine Darstellungungsform der globalen Mehrroboterpläne, bei der die räumliche Information reduziert wurde. In den Navigation Trees kann einfach gesehen werden, wie viele PRRs benötigt werden und wie viele TRRs benötigt werden, um von einem PRR zu einem benachbarten PRR zu gelangen. So stellt sich die Frage nach der Anzahl der benötigten Roboter als Traversierungsproblem auf den Navigation Trees. Hier konnte in Zusammenarbeit mit der Abteilung Informatik I der Universität Bonn ein Algorithmus gefunden werden, der in der Rechenzeit $O(nlog(n))$ die Anzahl der minimal benötigten Roboter berechnet. Der Algorithmus nutzt dabei das Prinzip der majorisierenden Kanten. Dies sind die Kanten im Baum, deren Gewicht größer ist als alle tiefer liegenden Kanten. Hier konnte gezeigt werden, dass diese majorisierenden Kanten in einer bestimmten Reihenfolge besucht werden müssen. Zudem wurde das zugrunde liegende Problem, die Traversierung von Graphen unter speziellen Bedingungen, verallgemeinert und gezeigt, dass auf allgemeinen Graphen dieses Traversierungsproblem in NP liegt. Die Ergebnisse wurden zusammen mit Dr. Elmar Langetepe und Andreas Lenerz in Brüggemann et al. [2013b] veröffentlicht.
Mit Hilfe der so gefundenen optimalen Besuchsreihenfolge können die globalen Mehrroboterpläne in Handlungsanweisungen übersetzt werden. Mit diesen Handlungsanweisungen kann ein Controller auf dem Roboter diesen entsprechend des globalen Mehrroboterplans steuern. Somit ist die Kette von der Planung zur Steuerung geschlossen. Die so gefundenen Handlungsanweisungen enthalten den abzufahrenden Weg sowie Anweisungen, wann welcher Roboter auf andere Roboter zu warten hat. Damit haben die Handlungsanweisungen sowohl spatiale als auch temporale Aspekte. Zudem können die Handlungsanweisungen zur Optimierung der Planausführung genutzt werden. Solange nicht ein spezieller Roboter benötigt wird, um die Handlungsanweisung auszuführen, können die Handlungsanweisungen zu bestimmten Zeitpunkten zwischen den Robotern getauscht werden. Dies ermöglicht verschiedene Koordinierungsmechanismen, wie z.B.

eine automatische Sortierung nach Geschwindigkeiten, bei der der langsamste Roboter die kürzeste Handlungsanweisung bekommt, oder das Vermeiden von Deadlocks. So ergibt sich neben den vereinfachten Planungsverfahren und der besseren Lesbarkeit ein weiterer Vorteil für die Einführung von globalen Mehrroboterplänen. Diese Ergebnisse sind in Brüggemann et al. [2012] veröffentlicht worden.

4.6 Zusammenfassung der Ergebnisse der globalen Navigationsplanung unter Nebenbedingungen

In diesem Kapitel wurde das globale Planungsproblem der koordinierten Navigation unter spatialen Nebenbedingungen behandelt. Dabei sind sogenannte globale Mehrroboterpläne erstellt worden. Diese Pläne repräsentieren Zielpunkte und Pfade für ein MRS. Bei der Repräsentation der Umgebung wurden einflussreiche Design-Ent-scheidungen getroffen. Dabei wurde auf ein Roadmap-Verfahren zurückgegriffen, welches die Umgebung in einem regelmäßigen Raster erfasst. Um sowohl die Bewegungs- als auch die Bedingungsinformationen darzustellen, wurden auf den dabei entstehenden Knoten zwei verschiedene Graphen definiert, der Bewegungsgraph und der Bedingungsgraph. Zur Vereinfachung der Vergleiche zwischen beiden, d.h. der Prüfung, ob ein Roboter zu einer Position fahren kann (Bewegungsgraph) und auch fahren darf (Bedingungsgraph), wurden die Seperated Connection Graphs (SCG) entwickelt.
Das Problem der koordinierten Navigation ist in zwei Teilprobleme zerlegt worden. Zunächst muss eine Zielkonfiguration gefunden werden, die den Startpunkt und die Zielpunkte enthält und dabei die Nebenbedingung nicht verletzt. Danach müssen Pfade für die Roboter gefunden werden, die zu dieser Zielkonfiguration führen, wiederum ohne die Nebenbedingung zu verletzen.
Der Fokus dieses Kapitels liegt dabei auf der Berechnung der Zielkonfiguration. Dazu konnte zunächst bewiesen werden, dass das Finden einer Endpositionierung und das Steinerbaumproblem äquivalent sind. Dies zeigt zudem, warum das allgemeine Problem der koordinierten Navigation von MRS unter spatialen Nebenbedingungen NP-hart ist. Die Tatsache, dass das Finden einer Zielkonfiguration sich auf das Finden eines Steinerbaumes reduzieren lässt, ermöglicht den ersten vorgestellten Algorithmus. Dieser ST-Planer dient in der weiteren Arbeit als Messbasis, da er auf dem Mehlhorn-Algorithmus für Steinerbäume basiert und dessen Eigenschaften gut bekannt sind.
Um die rein theoretische Betrachtung des Problems zu verlassen und eine praktische Lösung, die sich am Robotersystem orientiert, zu bieten, wurde der AgentPlaner eingeführt. Hierbei handelt es sich um ein lokales Such- und Optimierungsverfahren. Die Suchphase ist dabei von einem MPR-Flooding-Algorithmus inspiriert. Der AgentPlaner berücksichtigt dabei nicht nur, dass die Endkonfiguration gültig ist und wenige Knoten beinhaltet, er beachtet auch die Fahrmöglichkeiten der Roboter.
Um die Pfade, die zur Endkonfiguration führen, zu berechnen, ist eine Breitensuche

vorgeschlagen worden. Diese ist notwendig, da derzeit keine für A* gültige Heuristik angegeben werden kann, die auf der Basis von Bewegungs- und Bedingungsgraph die Entfernung zum Ziel abschätzen kann.
Die gewählte Repräsentation der Umgebung ermöglicht dabei einen ersten tieferen Einblick in das Problem, stellt zudem aber auch eine Einschränkung dar. Hier ist vor allem eine offene Frage, inwieweit die gefundenen Beziehungen z.B. zum Steinerbaum-Problem auch Algorithmen ermöglichen, die auf kontinuierlichen Umgebungen arbeiten können. So könnten Heuristiken, die für den euklidischen Raum Steinerbäume berechnen, auf dem Raum der Nebenbedingungen genutzt werden, vorausgesetzt dieser Raum lässt sich in vertretbarem Aufwand berechnen.
Insgesamt liefert das Kapitel einen Einblick in das neue Problem der koordinierten Navigation unter spatialen Nebenbedingungen. Verschiedenen Eigenschaften wurden gezeigt und bewiesen. Dabei ist der Zusammenhang zwischen Endkonfigurationen und Steinerbäumen besonders hervorzuheben. Dieser impliziert, dass auch bei einer anderen Repräsentation der Umgebung das Problem schwer ist. Um das Problem zu lösen, wurden zwei Algorithmen entwickelt, die auf unterschiedliche Art das Problem angehen. Erste Simulationsergebnisse zeigen die Nutzbarkeit dieser Algorithmen. Dabei zeigten sich schon unterschiedliche Eigenschaften die in Kapitel 5 weiter herausgearbeitet werden.
Bei der Planung der Endpositionierung wird in den derzeitigen Methode eine Endpositionierung bestimmt, die möglichst wenige Roboter benötigt. Dabei wird nicht eine vorhandene maximale Anzahl an Robotern berücksichtigt. Hier stellt sich die noch offene Frage, welche Punkte in einer Umgebung erreicht werden können, ohne die Nebenbedingung zu verletzten, wenn nur eine bestimmte Anzahl Roboter vorhanden ist. Gerade bei der Berücksichtigung von temporären Relais-Roboter ist diese Frage nicht trivial und bietet Möglichkeiten für weitere Untersuchungen und Anwendungen.
Zuletzt sei noch erwähnt, dass in dieser Arbeit bewusst eine statische Aufrechterhaltung der Nebenbedingung gewählt wurde. Das bedeutet, dass nach und nach eine Konfiguration aufgebaut wird, die die Nebenbedingung einhält. Dabei werden Roboter, die einmal ihren Zielpunkt erreicht haben, nicht mehr bewegt. Diese statische Aufrechterhaltung hat insbesondere bei so komplizierten Nebenbedingungen wie der Kommunikation den Vorteil, dass sie auch auf realen Systemen mit hoher Wahrscheinlichkeit aufrecht erhalten werden kann. Im Gegensatz zu dem hier gewählten statischen Ansatz kann auch eine dynamische Aufrechterhaltung der Nebenbedingung betrachtet werden. Dabei werden Roboter die gesamte Zeit so nachgeführt, dass die Nebenbedingung aufrecht erhalten wird. Es reicht also, nicht einmalig eine Endpositionierung zu bestimmen, sondern die Roboter müssen über die gesamte Ausführungszeit koordiniert bewegt werden. Es kann jedoch gezeigt werden, dass in bestimmten Situationen die statische Aufrechterhaltung beliebig viele Roboter mehr benötigt, als die dynamische Aufrechterhaltung der Nebenbedingungen. Details hierzu sind in Anhang A zu sehen.

5 Evaluation der Algorithmen

In diesem Abschnitt werden die Eigenschaften und Leistungen des STPlaner sowie des AgentPlaner und des FastAgentPlaner beleuchtet und verglichen. Da es in der Literatur keine bekannten Algorithmen gibt, die das gestellte Problem vollständig lösen, können beim Leistungsvergleich nur die hier entwickelten Algorithmen betrachtet werden. Dabei dient der STPlaner als Referenzalgorithmus. Aufgrund der großen Nähe zu den bekannten Steinerbaum-Heuristiken ist die Güte der Lösungen in Bezug auf Steinerbäume bekannt. So kann die Qualität der Endkonfigurationen des AgentPlaner und des FastAgentPlaner (FAP) besser eingeschätzt werden. Es zeigt sich bei der Auswertung, dass, obwohl die Endkonfigurationen des STPlaner weniger Roboter benötigen als die anderen Algorithmen, die Pläne, die aus den Endkonfigurationen entstehen, nur bedingt geeignet sind, auf einem Robotersystem ausgeführt zu werden, da die entstehenden Pfade sehr verschlungen sind und viele TRRs benötigt werden.

5.1 Datenerhebung zum Leistungsvergleich

Um einen Leistungsvergleich der drei vorgeschlagenen Algorithmen durchführen zu können, wurden alle Kombinationen von Umgebungen und Nebenbedingungen auf STPlaner, AgentPlaner und FastAgentPlaner getestet. Diese Versuche wurden mit verschiedenen Zielkonfigurationen wiederholt, um bestimmte Eigenschaften der einzelnen Algorithmen darstellen zu können. Im Weiteren wird zunächst der Versuchsaufbau dargestellt.

5.1.1 Gemessene Eigenschaften

Für den Vergleich der drei Verfahren

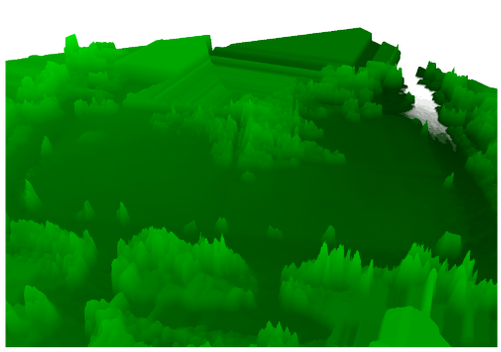
- STPlaner
- AgentPlaner
- FastAgentPlaner

wurden verschiedenen Eigenschaften der Pläne gemessen. Diese sind:

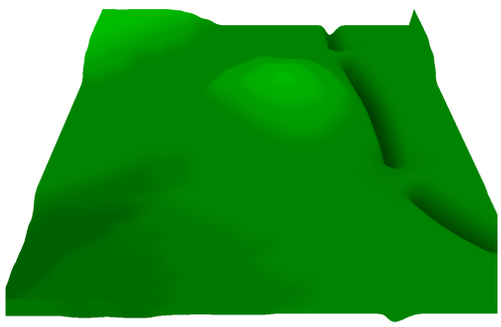
- Anzahl der Roboter in der Endpositionierung:
  Hierbei wurde die Anzahl der Roboter gemessen, die für die Endpositionierung benötigt werden. Dies sind die sogenannten permanenten Relais-Roboter. Dieser Wert gibt die Mindestanzahl an Robotern an, die zur Lösung des Navigationsproblems benötigt werden.
- Anzahl temporärer Roboterpositionen:
  Hier wird die Anzahl der temporären Roboter aufgeführt. Dieser Wert sagt etwas über die Komplexität des gefundenen Planes aus. Ein hoher Wert zeigt, dass die Ausführung des Planes schwierig ist.
- Zeit zur Berechnung der Endpositionierung:
  Hierbei wurde die Zeit in Sekunden gemessen, die der jeweilige Planer (STPlaner, AgentPlaner, FAP) für die Berechnung der Endkonfiguration benötigt hat.
- Zeit zur Berechnung des Plans:
  Aus der berechneten Endpositionierung muss noch der Pfad der Roboter berechnet werden. Dies geschieht zwar bei allen drei Planern mit dem gleichen Algorithmus, führt jedoch zu im Mittel unterschiedlichen Laufzeiten. Dies liegt daran, dass die Planer eine unterschiedliche Anzahl von SCGs während der Planung berechnen. Auf diese vorausberechneten SCGs kann bei der Pfadberechnung zurückgegriffen werden, was die Rechenzeit reduziert. Zusätzlich bedeuten mehr Roboter in der Endpositionierung und mehr temporäre Roboter eine erhöhte Rechenzeit.
- Längster Weg im Plan:
  Der Pfad, den die Roboter im globalen Mehrroboterplane nehmen müssen, bildet eine Baumstruktur. Der längste Weg vom Startpunkt aus zu einem Blatt ist der längste Weg, den ein Roboter zurücklegen muss. Dieser Wert wird als eine Messung der Güte des Planes genommen. Ist der längste Weg im Plan im Durchschnitt kurz, so ist auch die Ausführungsgeschwindigkeit des Planes kurz.
- Gesamtlänge des Weges:
  Hiermit wird die Gesamtgröße des Plans beschrieben. Dies ist ein Maß für die Komplexität des resultierenden Plans.

5.1.2 Umgebungen

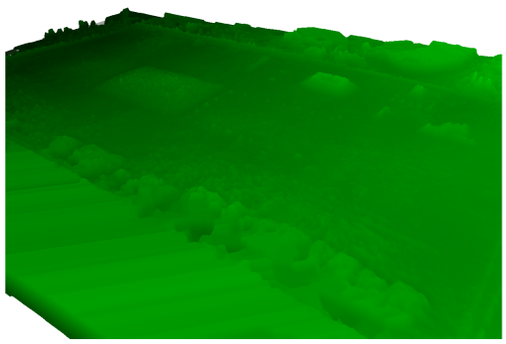
Um die Eigenschaften der einzelnen Algorithmen zu zeigen, wurden vier verschiedene Umgebungen betrachtet. Zwei dieser Umgebungen sind die simulierten Umgebun-



(a) Gebäude 3



(b) Landscape



(c) TDSuS

(d) Canyon

Abbildung 5.1: Die vier Simulationsumgebungen, die für die Evaluation genutzt wurden.

gen (Landscape und Canyon), die beiden anderen Umgebungen sind aus den Versuchen mit realen Robotersystemen entnommen und daher Karten von realen Umgebungen (Gebäude3 und TDSuS). Zur Gewinnung der realen Karten sind in Kapitel 7.3.2 weitere Details genannt.
Weitere Details zu den Umgebungen sind:

- Landscape (Abbildung 5.1b): 600 Knoten
  - Bewegungsgraph: 1945 Kanten
  - Bedingungsgraph (Distanz-Nebenbedingung 250m): 87551 Kanten
  - Bedingungsgraph (Sichtbarkeits-Nebenbedingung) 85314 Kanten
  - Bedingungsgraph (Kommunikations-Nebenbedingung) 60071 Kanten
- Canyon (Abbildung 5.1d): 610 Knoten
  - Bewegungsgraph: 2019 Kanten
  - Bedingungsgraph (Distanz-Nebenbedingung 250m): 89530 Kanten

  - Bedingungsgraph (Sichtbarkeits-Nebenbedingung) 81636 Kanten
  - Bedingungsgraph (Kommunikations-Nebenbedingung) 51190 Kanten
- TDSUS (Abbildung 5.1c): 714 Knoten
  - Bewegungsgraph: 1775 Kanten
  - Bedingungsgraph (Distanz-Nebenbedingung 250m): 86849 Kanten
  - Bedingungsgraph (Sichtbarkeits-Nebenbedingung) 46882 Kanten
  - Bedingungsgraph (Kommunikations-Nebenbedingung) 57292 Kanten
- Geb3 (Abbildung 5.1a): 222 Knoten
  - Bewegungsgraph: 612 Kanten
  - Bedingungsgraph (Distanz-Nebenbedingung 250m): 12383 Kanten
  - Bedingungsgraph (Sichtbarkeits-Nebenbedingung) 9725 Kanten
  - Bedingungsgraph (Kommunikations-Nebenbedingung) 9336 Kanten

5.1.3 Versuchsdurchführung

In jeder der Umgebungen wurde jeder Planungsalgorithmus mit den drei folgenden Nebenbedingungen getestet:

- Distanz-Nebenbedingung
- Sichbarkeits-Nebenbedingung
- Kommunikations-Nebenbedingung

Dabei wurden randomisierte Ausgangssituationen genutzt, d.h. folgende Parameter zufällig bestimmt:

- Startposition: Die Startposition wurde zufällig bestimmt.
- Anzahl Ziele: Die Anzahl der Zielpunkte wurde zwischen 1 und 10 Zielen zufällig bestimmt.
- Position der Ziele: Die Position jedes Ziels wurde zufällig bestimmt, allerdings so, dass sie auch vom Startpunkt aus erreichbar sind.

Damit wurden für jeden Durchlauf nur Szenarien gebildet, die auch lösbar sind. Jeder Algorithmus wurde in jeder Konfiguration (Umgebung / Nebenbedingung) 500-mal mit unterschiedlichen Ausgangssituationen getestet. So entstand eine breite Datenbasis für die Auswertung der Eigenschaften und für den Leistungsvergleich der Algorithmen.
Die Ergebnisse zeigen zunächst einen Leistungsvergleich der einzelnen Algorithmen zueinander. Ebenfalls können bestimmte Einflüsse der Umgebung auf die Leistung der Algorithmen abgelesen werden. Zudem kann durch die Aufschlüsselung der Ergebnisse nach Anzahl der zu erreichenden Ziele ebenfalls etwas über die Skalierung bezüglich der Anzahl Zielpunkte ausgesagt werden. Im Weiteren werden nun die einzelnen Messwerte ausgewertet, um zum Schluss jedem vorgeschlagenen Planer bestimmte Eigenschaften zuweisen zu können und zu zeigen, welcher Planer in welcher Situation zu bevorzugen

ist. Dabei werden hier jeweils Beispielhaft die Ergebnisse aller Planer auf allen Umgebungen für eine Nebenbedingung aufgezeigt. Die ausgewählten Ergebnisse gelten als repräsentativ für die Versuche.

5.2 Ergebnisse des Leistungsvergleichs

Im Folgenden werden die Ergebnisse der Versuch in Bezug auf die einzelnen Messgrößen dargestellt.

5.2.1 Anzahl der Roboter in der Endkonfiguration

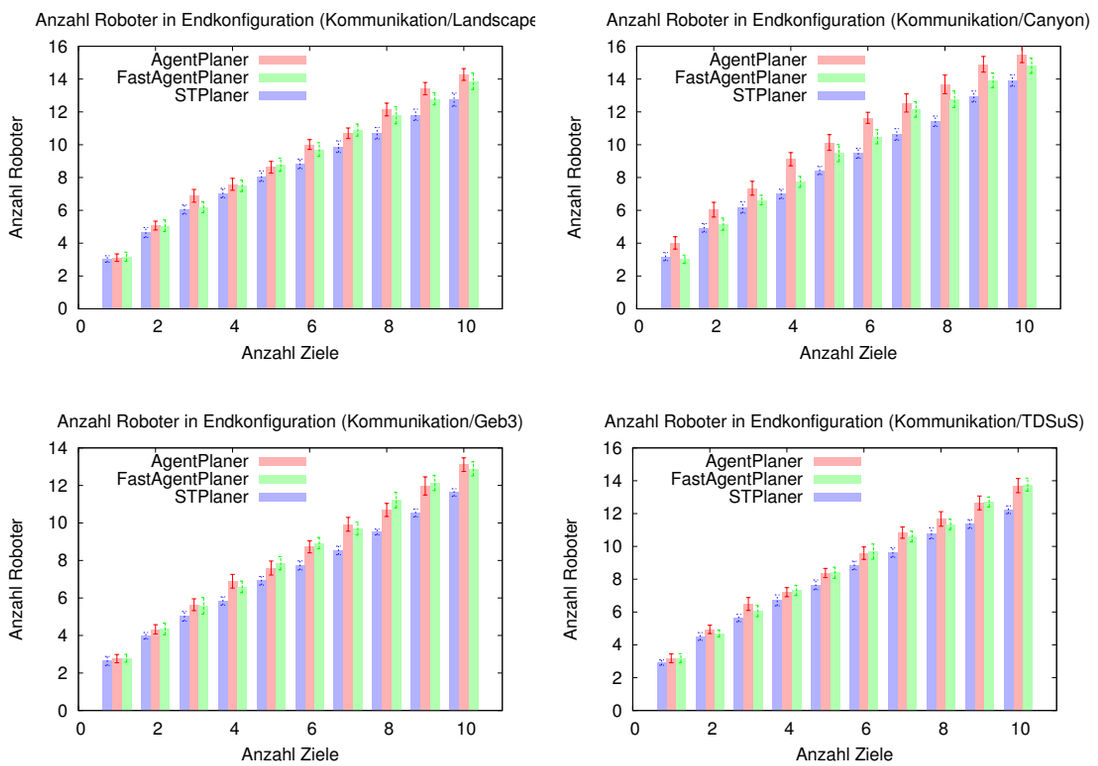


Abbildung 5.2: Anzahl der benötigten Roboter für die Endkonfiguration in unterschiedlichen Umgebungen unter der Kommunikationsnebenbedingung. Dabei wird die Anzahl benötigter Roboter nach Anzahl der zu erreichenden Ziele aufgeschlüsselt. Die Fehlerbalken geben das 95% Konfidenzintervall an.

Bei der Anzahl der Roboter in der Endkonfiguration erweist sich der STPlaner wie erwartet als bester Planer (siehe Abbildung 5.2). AgentPlaner und FAP sind, insbesondere bei Betrachtung der Varianz der Ergebnisse als gleichwertig einzustufen. Dabei zeigt es

sich, dass, verglichen mit dem STPlaner, diese agenten-basierten Verfahren nicht deutlich schlechter sind. So werden im Schnitt weniger als ein Roboter mehr benötigt. Die Gesamtanzahl benötigter Roboter ist bei den Umgebungen Landescape, Geb3 und TDSuS annähernd gleich und auch im Canyon nicht deutlich größer, so dass davon ausgegangen werden kann, dass die Umgebung keinen bemerkenswerten Einfluss auf die Anzahl der Roboter in der Endpositionierung hat. Das Wachstum der benötigten Roboter pro Zielpunkt ist linear und für alle Planer ähnlich.

Anzahl Roboter in Endkonfiguration									
	STPlaner			AgentPlaner			FastAgentPlaner		
	vis	com	dist	vis	com	dist	vis	com	dist
Canyon	**7,1**	8,7	**7,7**	8,1	**8,3**	8,5	8,1	9,6	8,7
Landscape	**7,3**	**8,2**	**7,5**	7,5	8,9	8,2	7,7	9,0	8,5
Geb3	**7,4**	**7,4**	**6,8**	8,3	7,7	7,7	8,3	8,2	7,7
TDSuS	**8,6**	**7,9**	**6,9**	10,3	8,5	7,9	9,3	9,1	8,1

Tabelle 5.1: Mittelwert der Anzahl Roboter in der berechneten Endkonfiguration aufgeschlüsselt nach Planer, Nebenbedingung und Umgebung. Hierbei ist der Mittelwert über alle 500 Versuche berechnet worden.

5.2.2 Anzahl der temporären Roboterpositionen

Auf die Anzahl der temporären Roboterpositionen haben nicht nur die verschiedenen Algorithmen Einfluss, sondern auch die Nebenbedingungen und die Umgebungen. Der AgentPlaner benötigt aufgrund der Suche auf den SCG-Nachbarschaften (siehe Kapitel 4) keine temporären Relaisroboter (siehe Abbildung 5.3). Die Anzahl der benötigten Roboter im globalen Mehrroboterplan entspricht also exakt der Anzahl Roboter in der Endpositionierung. Hier zeigt sich, dass schon etwa ein Roboter mehr in der Endkonfiguration die Nutzung von temporären Relaisrobotern überflüssig machen kann.
Im Gegensatz zum AgentPlaner benötigen der STPlaner und der FAP temporäre Relaisroboter. Hier sieht man jedoch, dass die Anzahl der benötigten TRRs deutlich streut. Dabei sind zwei Tendenzen zu erkennen. Zum einen der Einfluss der Umgebung, zum anderen der Einfluss des Nebenbedingung. So benötigen beide Algorithmen in der Canyon-Umgebung die meisten TRRs, danach folgt die TDSuS-Karte, dann Landscape und zum Schluss Gebäude3. Dies impliziert, dass die Umgebung eine Rolle dabei spielt, wie viele TRRs benötigt werden. In der Canyon-Umgebung mit ihren markanten Abbruchkanten gibt es viele Situationen, in denen zwischen zwei Positionen die Nebenbedingung erfüllt ist, jedoch der Weg, den der Roboter nehmen muss, viele TRRs enthält.
Auffällig ist auch die große Varianz der benötigten TRRs gerade in der Canyon und TDSuS Umgebung beim STPlaner. Je nach Positionierung der Ziele stellen sich in dieser Umgebungen also sehr unterschiedlich schwere Probleme.

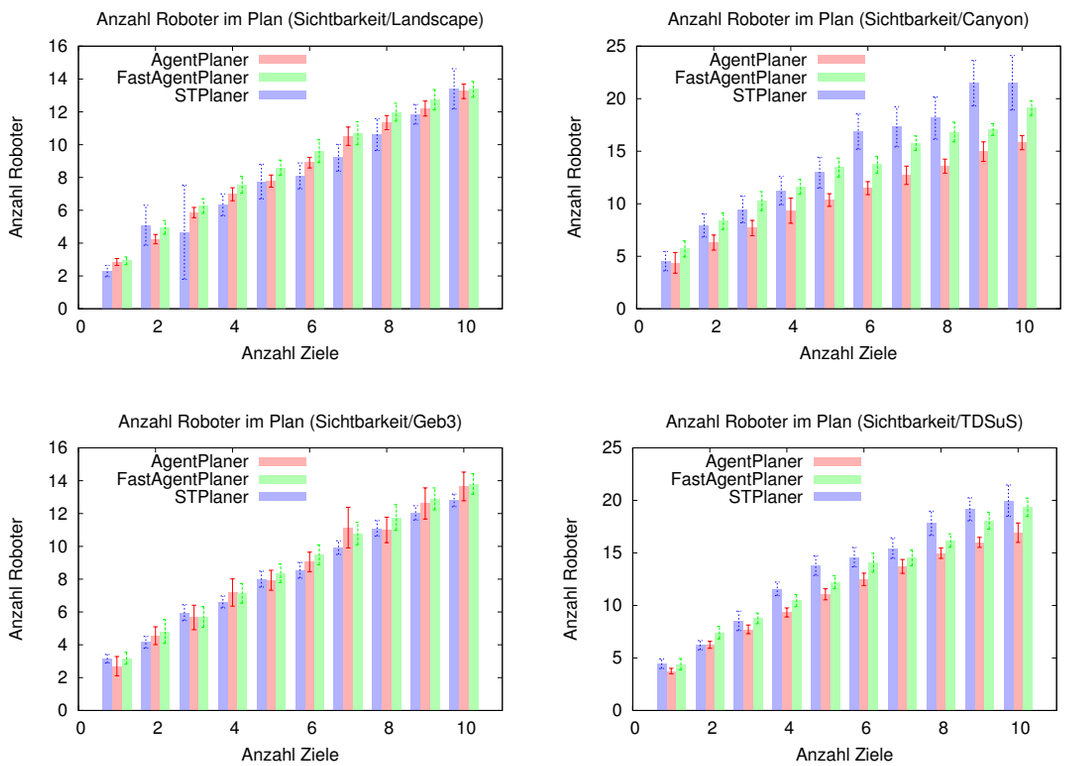


Abbildung 5.3: Anzahl der benötigten Roboter im globalen Mehrroboterplan (PRRs + TRRs) in unterschiedlichen Umgebungen unter der Sichtbarkeitsnebenbedingung. Dabei wird die Anzahl benötigter Roboter nach Anzahl der zu erreichenden Ziele aufgeschlüsselt. Die Fehlerbalken geben das 95% Konfidenzintervall an.

Neben dem gewählten Algorithmus und der Umgebung ist auch die Nebenbedingung ein Faktor für die Notwendigkeit von temporären Relaisrobotern. An den Zahlen in Tabelle 5.2 und in Abbildung 5.4 ist zu sehen, dass bei gleicher Umgebung und bei gleichem Algorithmus die Anzahl der TRRs für die einzelnen Nebenbedingungen sehr unterschiedlich ist. Dabei ist die Sichtbarkeits-Nebenbedingung diejenige Nebenbedingung, die im Schnitt die meisten TRRs erzeugt, gefolgt von der Kommunikations-Nebenbedingung und der Distanz-Nebenbedingung. Dies erklärt sich durch die Besonderheit der Sichtbarkeits-Nebenbedingung. Da keine Entfernungsbeschränkung für die Sichtbarkeit gegeben wurde, ist ein Knoten im Bedingungsgraph im Allgemeinen mit sehr vielen anderen Punkten verbunden. Allerdings sind diese Punkte nicht unbedingt Weg-zusammenhängend. Insbesondere Gräben stören die Sichtverbindung nicht. Dies wird bei der Distanz-Nebenbedingung ins Gegenteil verkehrt. Hier sind die Knoten, die im Bedingungsgraphen miteinander verbunden sind, nie weit voneinander entfernt. Zusätzlich bilden sie unabhängig von Hindernissen einen zusammenhängenden Bereich im Bedingungsgraphen.

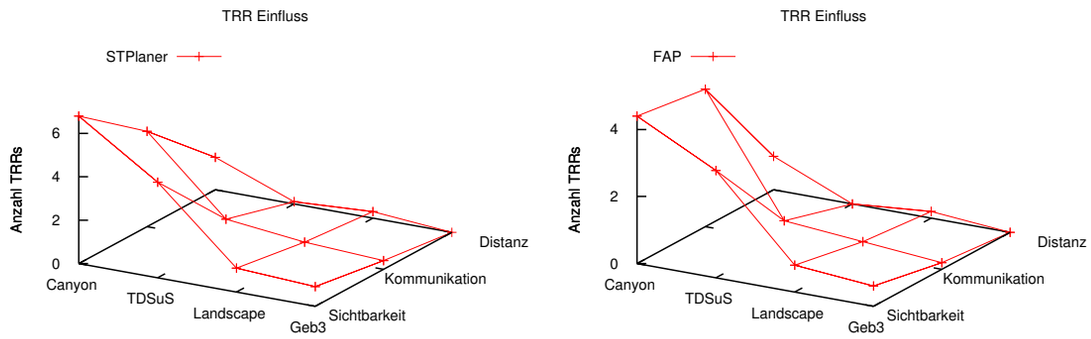


Abbildung 5.4: Visualisierung des Einflusses von Umgebung und Nebenbedingung auf die Anzahl der benötigten TTRs. Insgesamt zeigt sich, dass die Umgebungen und Nebenbedingungen nach ihren Kosten in Bezug auf benötigte TRRs sortiert werden können. Dabei ist Geb3 die einfachste (wenige TTRs) und Canyon die schwerste (viele TRRs) Umgebung. Unabhängig von der Umgebung ist die Distanznebenbedingung die leichteste und Sichtbarkeit die schwerste Nebenbedingung.

Anzahl benötigter temporärer Relaisroboter									
	STPlaner			AgentPlaner			FastAgentPlaner		
	vis	com	dist	vis	com	dist	vis	com	dist
Canyon	6,8	4,4	1,5	**0**	**0**	**0**	4,4	4,1	1
Landscape	1,1	0,6	0,3	**0**	**0**	**0**	0,8	0,4	0,2
Geb3	0,9	0,4	**0**	**0**	**0**	**0**	0,6	0,2	**0**
TDSuS	4,4	1,0	0,1	**0**	**0**	**0**	3,2	0,6	**0**

Tabelle 5.2: Mittelwert der Anzahl der zusätzlich benötigten Roboter (TRRs) in den berechneten globalen Mehrroboterplänen aufgeschlüsselt nach Planer, Nebenbedingung und Umgebung. Hierbei ist der Mittelwert über alle 500 Versuche berechnet worden.

5.2.3 Durchschnittliche Länge der längsten Pfade

Abbildung 5.5 zeigt die durchschnittliche Länge der längsten Pfade in den Plänen. Besonders auffällig ist hier, dass der AgentPlaner meist die kürzesten Pläne geniert. Das bedeutet, es ist zu erwarten, dass diese Pläne schneller ausgeführt werden können als die Pläne der anderen Algorithmen. Hier sind die Einflüsse der Umgebungen und der Nebenbedingung nicht so deutlich ausgeprägt wie bei der Frage nach der Anzahl der temporären Relaisroboter. Nur in der Kombination aus schwieriger Karte (TDSuS oder Canyon) mit komplexer Nebenbedingung (Sichtbarkeit) sind die Pläne des AgentPlaner deutlich besser als die Pläne der anderen Algorithmen; ansonsten sind die Unterscheide nur gering. Große Unterschiede in der Länge des längsten Pfades sind bei unterschiedlicher Anzahl an Zielen nicht festzustellen. Hier wird vor allem die Platzierung des

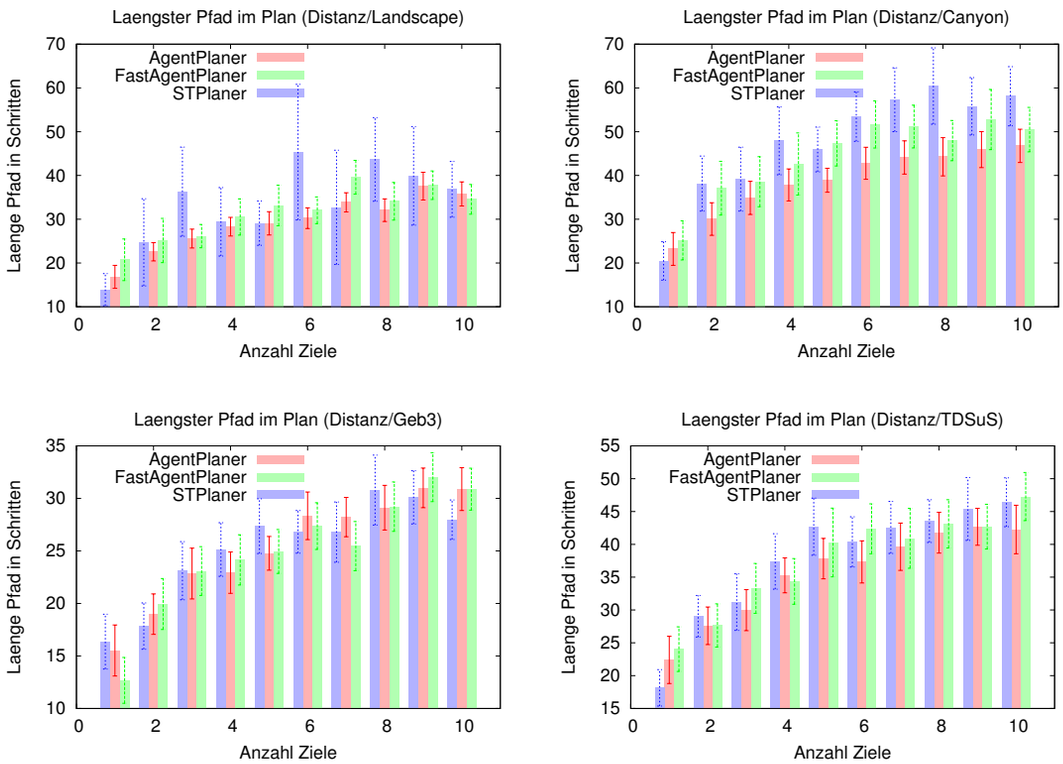


Abbildung 5.5: Längster Pfad im globalen Mehrroboterplan (in unterschiedlichen Umgebungen unter der Distanznebenbedingung. Dabei wird die Anzahl benötigter Schritte auf dem längsten Pfad nach Anzahl der zu erreichenden Ziele aufgeschlüsselt. Die Fehlerbalken geben das 95% Konfidenzintervall an.

am weitesten entfernten Ziel eine Rolle spielen. Dies erklärt auch die großen Varianzen der Pfadlänge.

Maximale Pfadlänge im globalem Mehrroboterplan									
	STPlaner			AgentPlaner			FastAgentPlaner		
	vis	com	dist	vis	com	dist	vis	com	dist
Canyon	57,9	63,3	47,8	**37,6**	**45,3**	**38,2**	64,4	62,5	45,0
Landscape	39,8	38,1	33,6	29,7	**28,8**	**28,7**	**29,4**	35,0	31,4
Geb3	27,2	28,8	**25,1**	**22,8**	**24,6**	25,5	24,8	26,9	**25,1**
TDSuS	64,3	49,3	37,3	**36,7**	**37,2**	**35,9**	64,0	45,6	37,7

Tabelle 5.3: Maximale Pfadlänge im globalem Mehrroboterplan aufgeschlüsselt nach Planer, Nebenbedingung und Umgebung. Hierbei ist der Mittelwert über alle 500 Versuche berechnet worden.

5.2.4 Zeit für die Berechnung der Endkonfiguration

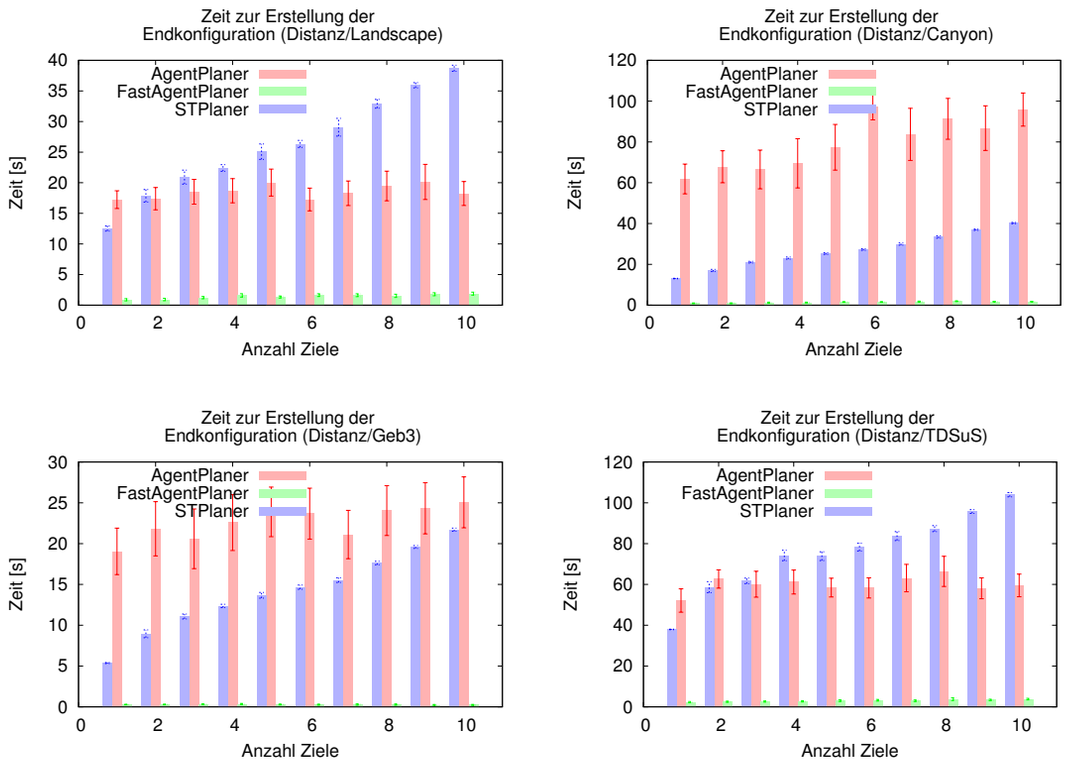


Abbildung 5.6: Zeit zur Berechnung einer Endkonfiguration in unterschiedlichen Umgebungen unter der Kommunikationsnebenbedingung. Dabei wird die Zeit in Sekunden nach Anzahl der zu erreichenden Ziele aufgeschlüsselt. Die Fehlerbalken geben das 95% Konfidenzintervall an.

Die Zeit für die Berechnung der Endkonfiguration (siehe Abbildung 5.6) berücksichtigt nur die eigentlichen Algorithmen STPlaner, AgentPlaner und FAP. Auch wenn solche Messungen von der konkreten Implementierung der einzelnen Algorithmen abhängen und damit nicht garantiert werden kann, dass die jeweils optimale Version gewählt wurde, können grundsätzliche Tendenzen erkannt werden. Die Versuche wurden jeweils auf einer Workstation durchgeführt (16 Kerne, 2,4 GH, 12GB Ram).
Auffällig ist hier vor allem, dass der FastAgentPlaner seinem Namen gerecht wird. Unabhängig von der Umgebung und den Nebenbedingungen wird die Endkonfiguration erheblich schneller gefunden als bei den anderen Verfahren. Für die beiden anderen Algorithmen kann kein genereller Trend beobachtet werden. Je nach Wahl der Umgebung und der Nebenbedingung kann sowohl der STPlaner als auch der AgentPlaner schneller als der jeweils andere sein. Dabei können die Geschwindigkeitsunterschiede deutlich ausfallen (wie z.B. in den Kombinationen Sichtbarkeit/Canyon, Sichtbarkeit/Landscape oder Distanz/Canyon) oder nur marginal sein.

Zudem zeigen die Versuche, dass die Anzahl der Ziele den AgentPlaner in der Laufzeit nicht beeinflusst. Während die durchschnittliche Zeit, die der STPlaner benötig, stetig mit der Anzahl der Ziele wächst, bleibt dies für den AgentPlaner in etwa konstant. So scheint der AgentPlaner für Probleme mit vielen Zielen deutlich geeigneter zu sein als der STPlaner.

Zeit für die Berechnung der Endkonfiguration [in Sekunden]									
	STPlaner			AgentPlaner			FastAgentPlaner		
	vis	com	dist	vis	com	dist	vis	com	dist
Canyon	54,0	13,0	27,1	15,1	32,0	79,0	**1,8**	**0,9**	**1,5**
Landscape	56,0	16,0	26,1	8,2	12,4	18,4	**0,4**	**1,1**	**1,4**
Geb3	24,5	7,8	14,1	2,0	12,2	22,7	**0,2**	**0,3**	**0,3**
TDSuS	16,8	29,9	74,8	29,7	19,0	60,3	**1,2**	**2,0**	**3,0**

Tabelle 5.4: Zeit für die Berechnung der Endkonfiguration aufgeschlüsselt nach Planer, Nebenbedingung und Umgebung. Hierbei ist der Mittelwert über alle 500 Versuche berechnet worden.

5.2.5 Gesamtzeit für die Planung

Die Gesamtzeit für die Planung ist in Abbildung 5.7 dargestellt. Hierbei handelt es sich um die Zeit, die benötigt wird, um den Plan vollständig zu erstellen. Das heißt, zur zeitlichen Komponente aus Abbildung 5.6 kommt noch die Dauer für die Berechnung des Pfades hinzu. Auch wenn der Algorithmus zur Berechnung des Pfades bei allen drei Planern gleich ist, so entstehen doch erhebliche Unterschiede. Diese Unterschiede basieren hauptsächlich auf zwei Faktoren:

- Länge und Komplexität des Weges: Einen Pfad zwischen zwei Relaisknoten zu finden dauert länger, je mehr TRRs für diesen Weg benötigt werden.
- Anzahl der vorausberechneten SCGs: Während bei der Berechnung der Endkonfiguration beim STPlaner und beim FAP keine SCGs berechnet werden, nutzt der AgentPlaner solche SCGs. Diese schon berechneten SCGs können dann bei der Pfadsuche genutzt werden.

Durch diese zwei Aspekte erklärt es sich, dass die Ergebnisse der gesamten Laufzeit sich durchaus von denen der reinen Konfigurations-Berechnung unterscheiden. Der zeitliche Vorteil des FAP ist nicht mehr so groß wie bei der Berechnung der Endpositionierung. Tatsächlich werden für die Wegberechnung in der Canyon-Umgebung in den Nebenbedingungen Kommunikation sowie Sichtbarkeit so viele SCGs benötigt, dass der Geschwindigkeitsvorteil aus der Endkonfigurations-Berechnung zunichte gemacht wird. In den Versuchen zeigt sich, dass nahezu alle möglichen SCGs auch berechnet werden. Da insbesondere in der Sichtbarkeitsnebenbedingung die SCGs viele Knoten beinhalten, ist ihre Berechnung insgesamt kostenintensiv. In den anderen Konfigurationen zeigt der FAP allerdings immer noch eine deutlich höhere Berechnungsgeschwindigkeit.

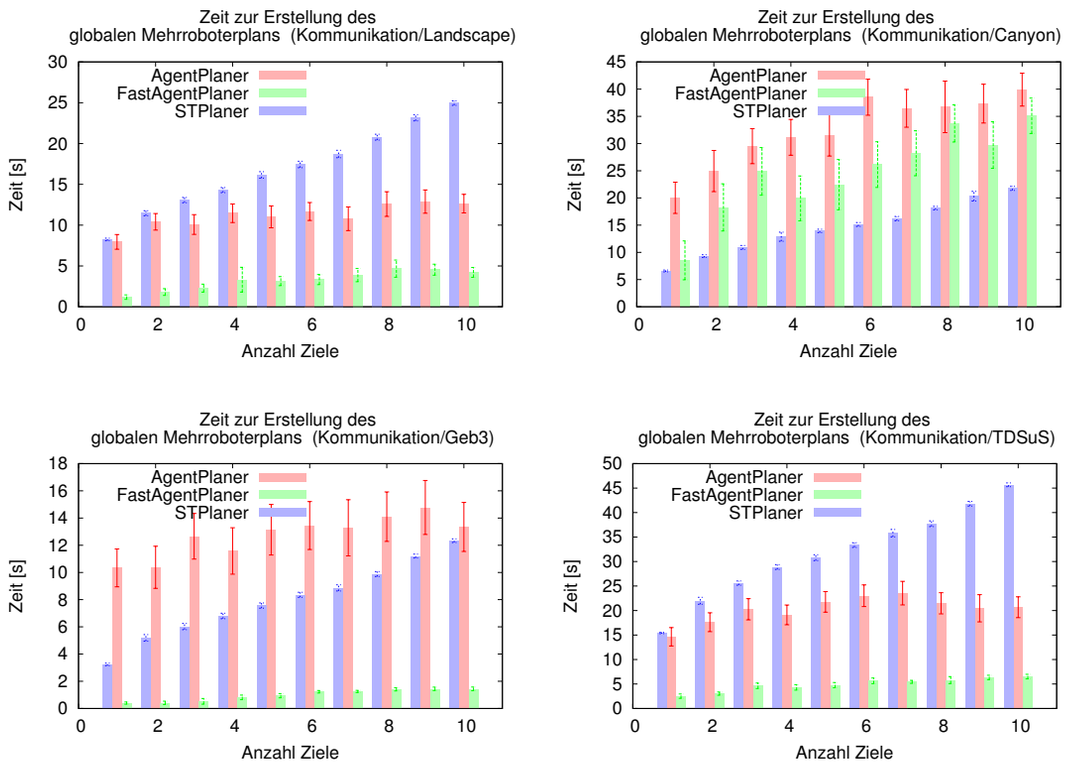


Abbildung 5.7: Zeit zur Berechnung des globalen Mehrroboterplans in unterschiedlichen Umgebungen unter der Kommunikationsnebenbedingung. Dabei werden die Zeit in Sekunden nach Anzahl der zu erreichenden Ziele aufgeschlüsselt. Die Fehlerbalken geben das 95% Konfidenzintervall an.

Auch hier zeigt sich, dass die Rechenzeit des AgentPlaner weitgehend unabhängig von der Anzahl der Ziele ist, jedoch eine große Streuung der Ergebnisse in den Versuchen zeigt. Dies liegt vor allem an der Platzierung der Ziele. Sind Ziele so nahe beieinander, dass ihre SCGs benachbart sind, so reichen oft schon wenige Expansionsschritte, um eine gültige Endpositionierung zu finden.

Allgemeine Eigenschaften

Mit den durchgeführten Tests können die drei vorgestellten Planungsalgorithmen wie folgt gegeneinander gestellt werden:

Zeit für die Berechnung des gesamten Plans [in Sekunden]									
	STPlaner			AgentPlaner			FastAgentPlaner		
	vis	com	dist	vis	com	dist	vis	com	dist
Canyon	57,7	**14,4**	28,5	**18,2**	32,5	80	108,8	24,6	**7,5**
Landscape	59,9	16,8	27,7	9,7	13,0	19,5	**3,1**	**3,2**	**4,6**
Geb3	24,7	8,2	14,7	2,4	12,6	23,5	**0,9**	**1**	**1,8**
TDSuS	18,6	31,2	77,9	30,1	20,2	63,2	**9,4**	**5,0**	**9,2**

Tabelle 5.5: Zeit für die Berechnung der globalen Mehrroboterpläne aufgeschlüsselt nach Planer, Nebenbedingung und Umgebung. Hierbei ist der Mittelwert über alle 500 Versuche berechnet worden.

	# Roboter in Endkonfiguration	# temporäre Roboter	Maximale Pfadlänge	Zeit Berechnung Endkonfiguration	Zeit Berechnung gesamter Plan
STPlaner	niedrig	hoch	hoch	mittel	mittel
Agenten Planer	mittel	keine	niedrig	mittel	mittel
FAP	mittel	hoch	hoch	niedrig	niedrig

5.3 Zusammenfassung der Ergebnisse der Evaluierung

Innerhalb der Evaluierung wurden STPlaner, AgentPlaner und FastAgentPlaner miteinander verglichen, um die Eigenschaften und Vorteile jedes einzelnen Algorithmus herauszustellen. Für diesen Vergleich ist insbesondere der Algorithmus STPlaner wichtig. Er dient hier als Vergleichswert für die beiden Agenten-basierten Verfahren. Der Steinerbaum-Planer ist, da er auf dem Algorithmus von Mehlhorn beruht, in der Laufzeit und in der Performance genau bekannt.
Um die Eigenschaften der Verfahren zu bestimmen, wurden vier verschiedene Umgebungen genutzt. Zwei davon waren Beispielumgebungen aus den Simulation, zwei weitere Umgebungen sind Karten von realen Umgebungen. Dabei wurden die Umgebungen so gewählt, dass sie unterschiedliche Eigenschaften abprüfen. So ist z.B. die Canyon-Umgebung mit ihren zwei Abbruchkanten insbesondere bei der Pfadplanung schwierig.
Anhand einer Vielzahl von Simulationsversuchen für jeden Algorithmus konnten unterschiedliche Leistungsdaten für die Algorithmen bestimmt werden. Zudem konnten die Eigenschaften, auf die bei der Entwicklung des jeweiligen Algorithmus geachtet worden ist, in den Tests bestätigt werden. Es zeigten sich folgende Ergebnisse.
Der STPlaner erweist sich als guter Vergleichsalgorithmus. Aufgrund der Nutzung von Steinerbaum-Algorithmen sind die Endpositionierung die dieser Algorithmus berechnet in den allermeisten Fällen die Konfigurationen mit den wenigsten PRRs. Um die Positionen der PRRs jedoch zu erreichen, werden im globalen Mehrroboterplan TRRs benötigt. Je nach Schwierigkeit der Umgebung (z.B. in der Canyon Umgebung) kann dies auch viele zusätzliche Roboter bedeuten. Da der STPlaner auf dem Mehlhorn Algorithmus

basiert, zeigt sich, dass er bei vielen Zielen eine längere Laufzeit hat, als bei wenigen Zielen. Durch den Aufbau von Distanzgraphen von einem Ziel zu den anderen erhöht sich die Laufzeit. Zudem ist die Laufzeit unabhängig von der Lage der Ziele in der Umgebung. Der STPlaner kann vor allem in den einfachen Umgebungen benutzt werden, da hier die wenigsten Roboter benötigt werden und zudem nur vereinzelt TRRs eingesetzt werden müssen.
Der AgentPlaner benötigt im Schnitt etwa 1 - 2 PRRs mehr als der STPlaner. Bemerkenswert daran ist aber, dass die geringfügige Erweiterung der Roboteranzahl ausreicht, um einen globalen Mehrroboterplan zu erstellen, der keine weiteren TRRs benötigt. Damit sind nicht nur die Pläne des AgentPlaner einfacher auszuführen, auch das Problem, wie viele Roboter insgesamt benötigt werden (siehe Abschnitt 4.5) stellt sich nicht und die Handlungsanweisungen können einfach berechnet werden. Auch wenn hier die Ergebnisse nicht eindeutig sind, sind die Pläne des AgentPlaner häufig kurz, so dass auch die Ausführungsgeschwindigkeit dieser Pläne hoch sind. Bei der Laufzeit zur Berechnung der Endpositionierung sowie des gesamten globalen Mehrroboterplans fällt auf, dass die Rechenzeit nicht abhängig von der Anzahl der zu erreichenden Ziele ist. Hier ist die Laufzeit relativ konstant. Der STPlaner eignet sich daher vor allem in den komplexen Umgebungen, da die resultierenden Pläne trotzdem nur wenige Roboter (keine TRRs) benötigen und von realen Mehrrobotersystemen leicht auszuführen sind.
Der FastAgentPlaner zeigt sich in seinen Eigenschaften als Mischung der beiden vorhergehenden Algorithmen. Dabei erbt er zunächst die schlechten Eigenschaften von STPlaner und AgentPlaner. Der FastAgentPlaner benötigt in etwa so viele PRRs in den Endpositionierung wie der AgentPlaner und damit mehr als der STPlaner. Allerdings benötigt er zudem TRRs, um die Roboter an die Positionen der PRRs zu bringen. Zudem sind die resultierenden Pfade ähnlich komplex und ähnlich lang wie die des STPlaner. Allerdings gewinnt der FastAgentPlaner dadurch an Rechengeschwindigkeit, so dass gerade Ergebnisse einer Endpositionierungmindestens eine Größenordnung schneller zu erwarten sind. So eignet sich der FAP vor allem dann, wenn z.B. Positionen für Relaisknoten berechnet werden sollen oder viele Anfragen an Endpositionierung durchgeführt werden sollen.

6

Reaktion auf veränderliche Umgebungen

Bisher ist bei allen Überlegungen davon ausgegangen worden, dass die Umgebung im Voraus vollständig bekannt ist und sich auch während der Ausführung der Pläne nicht ändert. Diese Annahme war notwendig, um die grundsätzlichen Probleme und Eigenschaften der koordinierten Navigation unter räumlichen Nebenbedingungen zu erkennen und zu beschreiben. Bei realen Einsätzen der Planungssoftware ist aber davon auszugehen, dass die vorhandenen Karten zumindest in Details nicht mehr mit der Realität übereinstimmen. Solche Details können für die Planausführung irrelevant sein oder aber den gesamten Plan ungültig werden lassen. So schafft z.B. eine blockierte Brücke, die laut Plan benutzt werden soll, einen Fall, bei dem der Plan nicht sinnvoll weiterzuführen ist.

Änderungen in der Umwelt können dabei nicht nur die Bewegung des Roboters beeinflussen, sondern auch die Nebenbedingung dergestalt verändern, dass der aktuelle Plan ungültig wird. In diesem Kapitel werden zwei Arten von veränderlichen Umgebungen untersucht. In Kapitel 6.1 werden dynamische Umgebungen betrachtet. Hierbei ist der Status des Graphen jederzeit bekannt und es muss auf Veränderungen reagiert werden. Kapitel 6.2 behandelt die Planung auf unbekannten Umgebungen. Dabei ist der Status der Graphen zunächst unbekannt und muss erkundet werden. Die Änderungen an den globalen Mehrroboterplänen müssen sich auch in den Handlungsanweisungen wiederfinden. Wie Handlungsanweisungen dynamisch angepasst werden, wird in Kapitel 6.3 betrachtet.

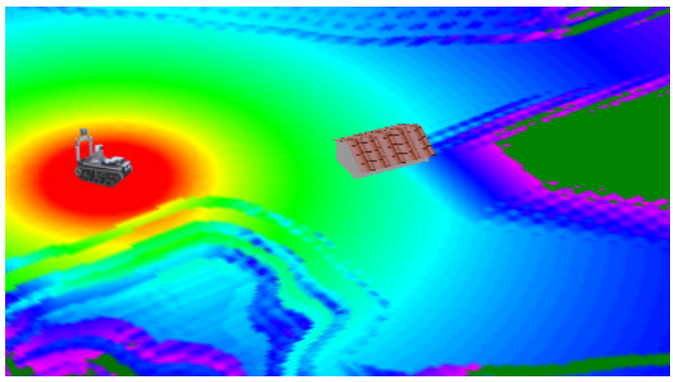


Abbildung 6.1: Vorher unbekannte Hindernisse beeinflussen ebenfalls die Nebenbedingung. Hier wird die Abschattung der Funkreichweite an einem Haus gezeigt. Ohne die vorherige Kenntnis des Hauses müsste angenommen werden, dass dahinter weiterhin Kontakt zu anderen Robotern bestehen kann.

6.1 Erscheinen von unbekannten Hindernissen

Zunächst sollen dynamische Umgebungen betrachtet werden. Dabei wird als dynamische Umgebung eine Umgebung definiert, die zwar vor der Berechnung des Plans bekannt ist, in der aber während der Ausführung des Plans Hindernisse auftauchen oder verschwinden können. Innerhalb dieser Arbeit wird nur der Fall betrachtet, dass zusätzliche Hindernisse auftauchen. Nur dieser Fall kann einen existierenden Plan ungültig werden lassen. Das Verschwinden eines Hindernisses kann deutlich kürzere globale Mehrroboterpläne ermöglichen, die Kosten der Neuplanung rechtfertigen die Suche nach solchen Optimierungen meist jedoch nicht. Wird nun ein Hindernis in der Umwelt detektiert, so muss es in der Karte eingetragen werden. Wie in Abbildung 6.1 werden auch in allen weiteren Abbildungen solche Hindernisse exemplarisch als Haus oder eine Gruppe von Häusern dargestellt.
Mit dem Eintrag in die Karte müssen sowohl der Bewegungsgraph als auch der Bedingungsgraph überprüft und gegebenenfalls angepasst werden. Für den Bewegungsgraphen ist dies einfach, da die Grenzen des Hindernisses auch die Grenzen dessen Einflusses auf den Bewegungsgraphen darstellen. Für den Bedingungsgraphen gilt dies nicht. Hier können auch Knoten beeinflusst werden, die weit von dem Hindernis entfernt liegen. Bisher existiert noch keine Methode, mit der der Bedingungsgraph schnell an die veränderte Umgebung angepasst werden kann. Daher wird der Bedingungsgraph nach einer Veränderung der Umgebung komplett neu aufgebaut. Dies stellt den aufwendigsten Schritt in der Berechnung von dynamischen Umgebungen dar.
Durch die Änderung an Bewegungs- und Bedingungsgraph ändern sich auch die berechneten SCGs. Um nicht, wie beim Bedingungsgraph, die SCGs komplett neu zu berech-

nen, wird vorgeschlagen, dass für SCGs in dynamischen Karten zusätzlich gespeichert wird, von welchen Knoten sie bei der Berechnung abhängen. Ändert sich nun durch das Einfügen des Hindernisses eine Kante an solch einem Knoten, gilt der dazugehörige SCG als nicht mehr aktuell und muss bei Bedarf neu erstellt werden.
Dieses Vorgehen hat den weiteren Vorteil, dass nach dem relativ aufwendigen Update-Schritt der Umgebung sehr einfach bestimmt werden kann, ob der aktuelle Plan noch gültig ist. Sind die SCGs, aus denen der Plan besteht, noch aktuell, hat das Hindernis keinen Einfluss auf den ursprünglichen Plan und er kann weiterhin ausgeführt werden. Sind allerdings ein oder mehrere SCGs nicht mehr aktuell und müssen neu berechnet werden, kann dies auf einen ungültigen Plan hindeuten. In diesem Fall müssen weitere Tests vorgenommen werden.
Sind SCGs von der Änderung betroffen, die auch im globalen Mehrroboterplan genutzt werden, kann die Gültigkeit des Plans mit folgendem mehrstufigen Verfahren getestet werden:

1. Test auf Gültigkeit der Endpositionierung: Dabei wird überprüft, ob die Endpositionierung noch aus einer Zusammenhangskomponente besteht.
2. Test auf Gültigkeit der temporären Relaispunkte: Dieser Schritt ist nur notwendig, wenn der Plan temporäre Relais-Roboter enthält. Dann wird überprüft, ob die existierenden Relaisketten noch miteinander verbunden sind.
3. Test auf Gültigkeit des Pfades: Der berechnete Pfad zwischen zwei SCGs muss überprüft werden.

Dabei ist zum einen zu beachten, dass nur die SCGs überprüft werden müssen, die durch das Hindernis beeinflusst wurden. Zum anderen kann der Test abgebrochen werden, sobald der erste Teiltest negativ ausfällt.
Wie nun mit einem ungültigen Plan weiter verfahren wird, hängt von dem Algorithmus ab, mit dem er vorher erzeugt wurde. So sind die Ergebnisse, die der STPlaner zuvor berechnet hat, nun nicht mehr aktuell. Bei Beibehaltung des derzeitigen Konzepts des STPlaner muss die Berechnung einer Endpositionierung und der daraus folgenden Pfade und Handlungsanweisungen neu durchgeführt werden. Dabei kann, auch wenn das Hindernis nur einen kleinen Teil des alten Planes beeinflusst hat, ein völlig anderer Plan entstehen. Eine Möglichkeit, die völlige Neuberechnung des unterliegenden Steinerbaums des STPlaner zu umgehen, könnten Algorithmen für dynamische Steinerbäume wie z.B. in Blin et al. [2009] oder Ding and Ishii [2000] bieten. Solch eine Lösung ist jedoch nicht innerhalb des Fokus dieser Arbeit.
Im Gegensatz zum STPlaner trifft der AgentPlaner nur lokale Entscheidungen und ist daher dafür geeignet, auch in dynamischen Umgebungen angewendet zu werden. Wie oben dargestellt, wird ein vorher unbekanntes Hindernis auf der Karte eingezeichnet. Durch dieses Hindernis verändern sich G_{Bed} und G_{Bew} und damit die daraus berechneten SCGs. Um zu analysieren, wie der AgentPlaner auf diese Änderungen reagiert, werden die drei oben genannten Tests zugrunde gelegt. Dabei müssen nur Test 1 und

Test 3 durchgeführt werden, da der AgentPlaner grundsätzlich keine temporären Relais-Roboter benötigt.
Für den Fall, dass die Endpositionierung nicht mehr gültig ist, muss der Planer neu planen. Im Falle des AgentPlaner bedeutet dies, dass der MPR-Baum, der die Endpositionierung darstellt, nicht mehr aus einer Zusammenhangskomponente besteht. Dies heißt, dass nicht mehr alle aktiven Agenten miteinander verbunden sind. Hier ist also die Abbruchbedingung für die Expansion der Agenten nicht mehr gegeben. Daher werden die Agenten neu expandieren, bis die Abbruchbedingung wieder erfüllt ist.
Diese neue Expansion unterscheidet sich von der ersten (initalen) Expansion zur Suche einer Endkonfiguration in der Anzahl der aktiven Agenten. Während bei der initialen Suche nur die Ziele und der Startpunkt aktiv waren, sind nun auch die Agenten aktiv, die vorher Relaispunkte dargestellt haben. Damit hat der Algorithmus zu diesem Zeitpunkt mehr Informationen über die Umgebung als zur ersten Suche. Im Prinzip muss der AgentPlaner also nur wieder die Lücke schließen, die durch die Umgebungsänderung entstanden ist. Ist diese Lücke z.B. dadurch zu schließen, dass nur ein Roboter anders platziert werden muss, so wird dies schnell gefunden und der restliche Plan ändert sich nicht. Bei notwendigen größeren Änderungen (z.B. wenn eine Brücke gesperrt ist) können auch größere Änderungen am Plan vorgenommen werden.
Schlägt der dritte Test fehl, kann ein neuer Weg zwischen den einzelnen Relaisknoten mittels Dijkstra-Algorithmus berechnet werden. Dass ein solcher Weg existiert, ist sichergestellt, da der Test 1 bestanden wurde. Eine Beispielplanung für eine dynamische Umgebung ist in Abbildung 6.2 zu sehen.

6.2 Planung in unbekannten Umgebungen

Im Gegensatz zu den dynamischen Umgebungen ist bei unbekannten Umgebungen keine Karte zur initalen Planung vorhanden. Erst durch das Bewegen der Roboter in der Umgebung wird diese erkundet und damit bekannt. Dabei wird angenommen, dass die Roboter immer ein bestimmtes Umfeld um sich herum wahrnehmen können. Zudem wird angenommen, dass einmal erkundete Bereiche sich nicht mehr ändern können. Durch das Setzten des Startpunktes ist der Bereich um den Startpunkt bekannt, da hier die Roboter starten (vergleiche Abbildung 6.3). Da zu erwarten ist, dass sich Bedingungs- und Bewegungsgraph mit jedem explorierten Teilstück ändern, wird hier mit der gleichen Begründung wie bei den dynamischen Umgebungen nur der AgentPlaner betrachtet.
Da die Umgebung zunächst unbekannt ist, müssen zur Berechnung des Bewegungs- und Bedingungsgraphen einige Annahmen getroffen werden:

- Unbekannte Bereiche sind flach und ohne Hindernisse: Diese Annahme beeinflusst sowohl den Bewegungsgraphen wie auch den Bedingungsgraphen. Der Bewegungsgraph ist in unbekannten Umgebungen gleichmäßig und zwischen den benachbarten Knoten vollverknüpft. Für den Bedingungsgraphen hat diese An-

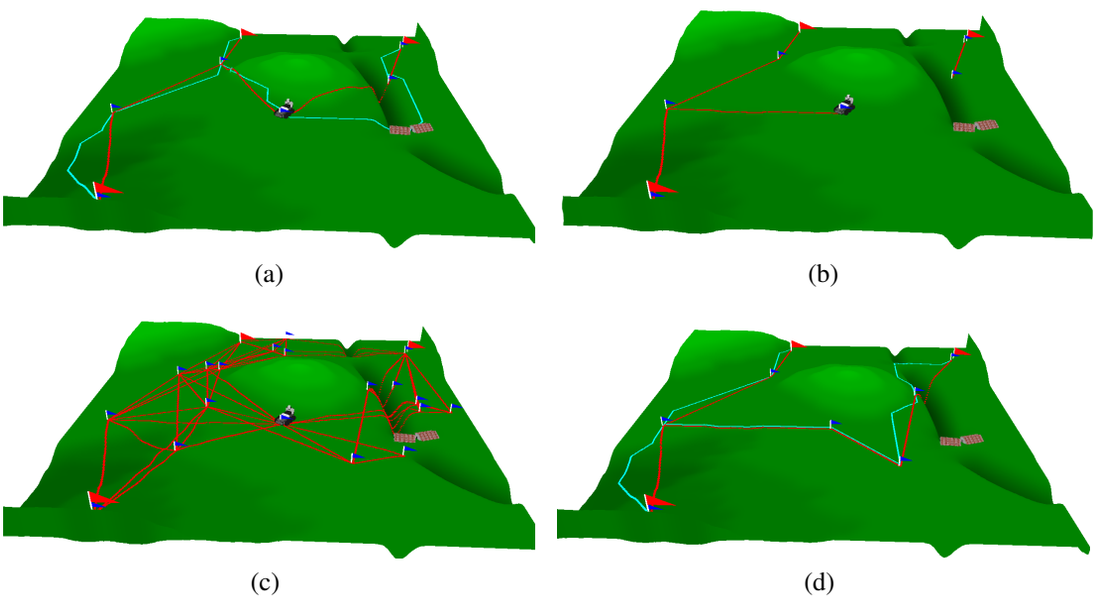


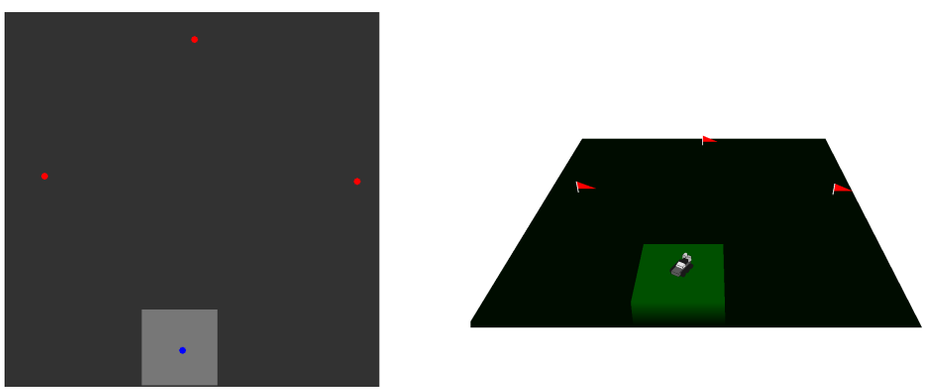
Abbildung 6.2: Beispielplanung des AgentPlaner beim Auftauchen eines unerwarteten Hindernisses. a) Initaler Zustand beim Entdecken des Hindernisses (hier als Haus dargestellt). Der ursprüngliche Plan (türkis) führte über die untere Brücke. Dieser Weg wird nun durch ein Hindernis unpassierbar. b) Durch die Änderungen des Bedingungs- und Bewegungsgraphen zerfällt die Endkonfiguration in zwei Zusammenhangskomponenten. Damit ist die Abbruchbedingung, die die Agenten vom Expandieren abhält, nicht mehr gültig. c) Ausgehend von den schon aktivierten Agenten (Startpunkt, Ziel- und Relaispunkte) expandieren die Agenten erneut. Durch die Nutzung der Relaisknoten startet die Suche nicht neu, sondern mit Vorwissen. d) Durch dieses Vorwissen wird der Teil des Planes, dessen Umgebung sich nicht geändert hat, kaum modifiziert. Der rechte Teil des Planes wird dagegen so angepasst, dass der Pfad jetzt die zweite, obere Brücke benutzt und damit wieder eine gültige Endkonfiguration darstellt.

nahme unterschiedliche Auswirkungen, je nachdem welche Bedingung gewählt wurde. So ist z.B. für die Sichtbarkeitsbedingung der Bedingungsgraph vollverknüpft.

- Wenn bei der Berechnung der Nebenbedingung C(a,b)=$\{0,1\}$ einer der Punkte a oder b in unbekanntem Gebiet liegt, wird angenommen, dass der unbekannte Punkt die gleiche Höhe hat wie der bekannte Punkt: Dies ist insbesondere für den Bedingungsgraphen von Bedeutung. Gerade für die Sichtbarkeitsbedingung und die Kommunikationsbedingung ist es wichtig, auf welcher Höhe die Punkte a und b liegen. Sind dazwischen Punkte, die höher liegen, so wird bei der Kommunikation die Signalstärke gedämpft und die Sichtbarkeitsbedingung ist nicht erfüllt. Die Annahme, dass beide Punkte dieselbe Höhe haben, ist dabei konservativ. Im Allgemeinen wird dadurch die Anzahl der Kanten im Bedingungsgraph unterschätzt, d.h. bei Erkundung der Karte werden hauptsächlich Kanten eingefügt und nur ver-

einzelt entfernt.

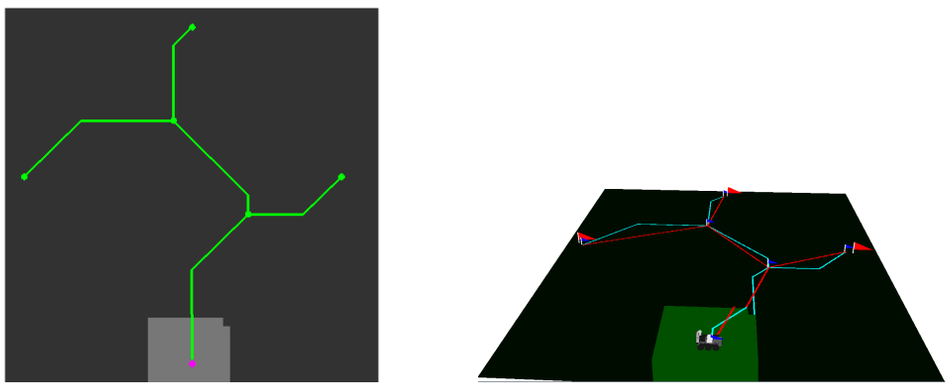
- Liegen beide Punkte bei der Berechnung der Nebenbedingung C(a,b)=$\{0,1\}$ in unbekannter Umgebung, so wird für beide Punkte eine Standardhöhe angenommen: Auch hier wird eine konservative Schätzung vorgenommen, die die möglichen Verbindungen normalerweise unterschätzt.
- An der Grenze zwischen bekannter und unbekannter Umgebung existiert ein stetiger Übergang: Diese Annahme ist notwendig, da sonst der Bewegungsgraph an den Grenzen abbrechen würde. Damit würde der Bewegungsgraph zumindest in zwei Komponenten zerfallen, der bekannten und der unbekannten Umgebung, womit jeder Plan, der Relaisknoten in unbekannter Umgebung hat, von vornherein unerfüllbar wäre.



(a) (b)

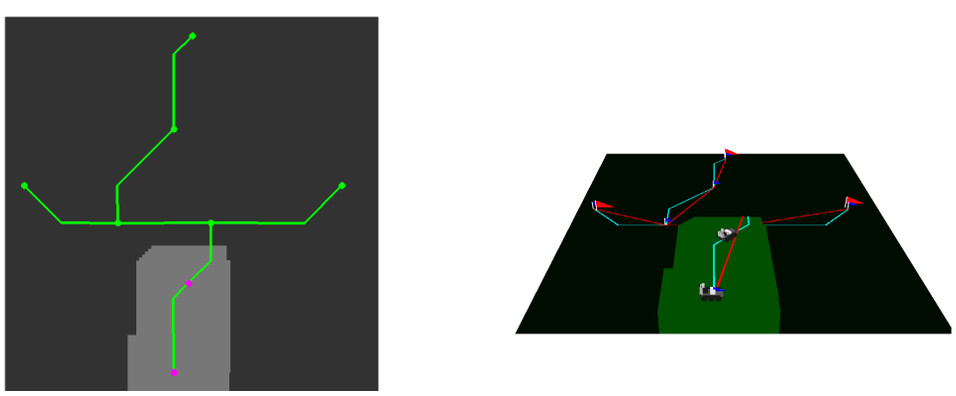
Abbildung 6.3: Links: 2D-Darstellung der Startsituation in unbekannter Umgebung. Blau ist dabei die Startposition, rot die Zielpositionen. Die Umgebung um die Startposition herum ist bekannt, da dort die Roboter stehen. Die restliche schwarze Fläche ist unbekannt. Rechts: Die gleiche Situation in der 3D-Darstellung.

Aufgrund der oben genannten Annahmen kann der AgentPlaner die Planung, wie im Kapitel 4.2.2 beschrieben, durchführen. Die initiale Planung ist in Abbildung 6.4 gezeigt. Mit der Erstellung des Plans werden die Handlungsanweisungen für die Roboter berechnet. Mit diesen Handlungsanweisungen können dann die Roboter vorwärts bewegt werden. Bei der Bewegung werden sukzessive weitere Teile der Umgebung erkundet (siehe Abbildung 6.5).
Durch das Erkunden neuer Teile der Umgebung muss auch der Bewegungs- und Bedingungsgraph angepasst werden. Durch die Änderung der Graphen kann der aktuelle Plan als nicht mehr gültig erkannt werden. Für diese Erkennung werden die gleichen Prüfungsverfahren wie bei den dynamischen Umgebungen genutzt. Dabei wird zunächst getestet, ob die aktiven Agenten nach der Aktualisierung benachbart sind. Ist dies der



(a) (b)

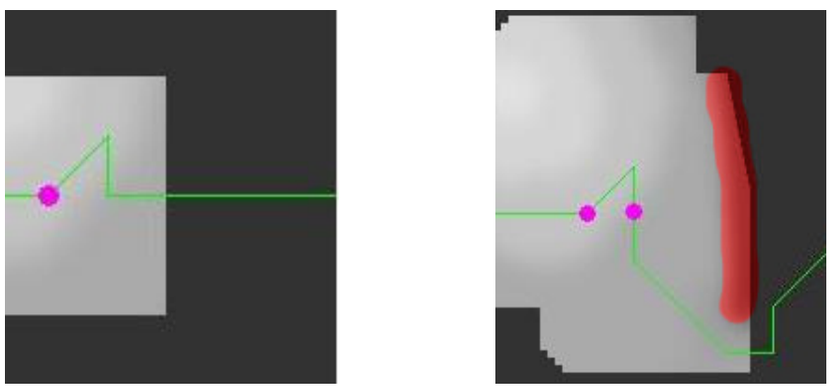
Abbildung 6.4: Initiale Planung in unbekannten Umgebungen. Links die 2D-Draufsicht und rechts das 3D-Modell. Da die unbekannte Umgebung als flach angesehen wird, ist der Plan darauf sehr einfach.



(a) (b)

Abbildung 6.5: Situation nach den ersten Bewegungen der Roboter. Durch das Abfahren des Planes werden weitere Teile der Umgebung sichtbar.

Fall, so werden noch die Pfade auf ihre Gültigkeit überprüft. Wenn die Agenten nicht mehr benachbart sind, ist die Abbruchbedingung für die Agentenexpansion nicht mehr gegeben, und der Expansionsschritt wird erneut gestartet. Durch die Expansion wird ein neuer MPR-Baum gefunden, der zu einem neuen globalen Plan führt. Sind die Agenten zwar noch verbunden, aber der Pfad nicht mehr gültig, so kann ein neuer Pfad berechnet werden. Dass ein solcher Pfad existiert, ist dadurch sichergestellt, dass die Agenten benachbart sind. Ein Beispiel für eine solche Änderung des Pfades ist in Abbildung 6.6 zu sehen. Hierbei kann zudem ein Problem beobachtet werden, welches im folgenden Unterkapitel genauer beschrieben wird.



(a) (b)

Abbildung 6.6: Erkennen eines ungültigen Pfades (Detailansicht eines globalen Mehrroboterplans). a) Es ist ein noch nahezu gerader Pfad (grüne Linie) zum Zielpunkt geplant. b) Ein Roboter bleibt als PRR stehen, der zweite Roboter bewegt sich vorwärts (Roboter in magenta). Nun zeigt sich der Graben (rot unterlegt) und der Weg wird dementsprechend angepasst.

6.2.1 Stabilisierung der globalen Mehrroboterpläne

Grundsätzlich treffen bei der koordinierten Navigation unter Nebenbedingung auf unbekannten Umgebungen zwei gegensätzliche Aufgaben aufeinander. Zum einen sollen die vorgegebenen Zielpunkte schnell erreicht werden, zum anderen ist die Umgebung unbekannt und muss erst nach einem möglichen Weg abgesucht werden. Somit konkurrieren die Aufgaben der schnellen Navigation und der Exploration miteinander. Durch die Nutzung des AgentPlaner ist die Komponente „Exploration“ jedoch nicht explizit adressiert, sondern wird als Nebeneffekt beobachtet. Dementsprechend ist in der Standardversion des AgentPlaner eine häufige Umplanung zu erkennen. Diese Umplanung entsteht dadurch, dass weitere Bereiche bekannt werden und durch die Anpassung des Bewegungs- bzw. Bedingungsgraphen der aktuelle Plan nicht mehr gültig ist. Ein Beispiel für diese häufige Umplanung ist in Abbildung 6.7 zu sehen.

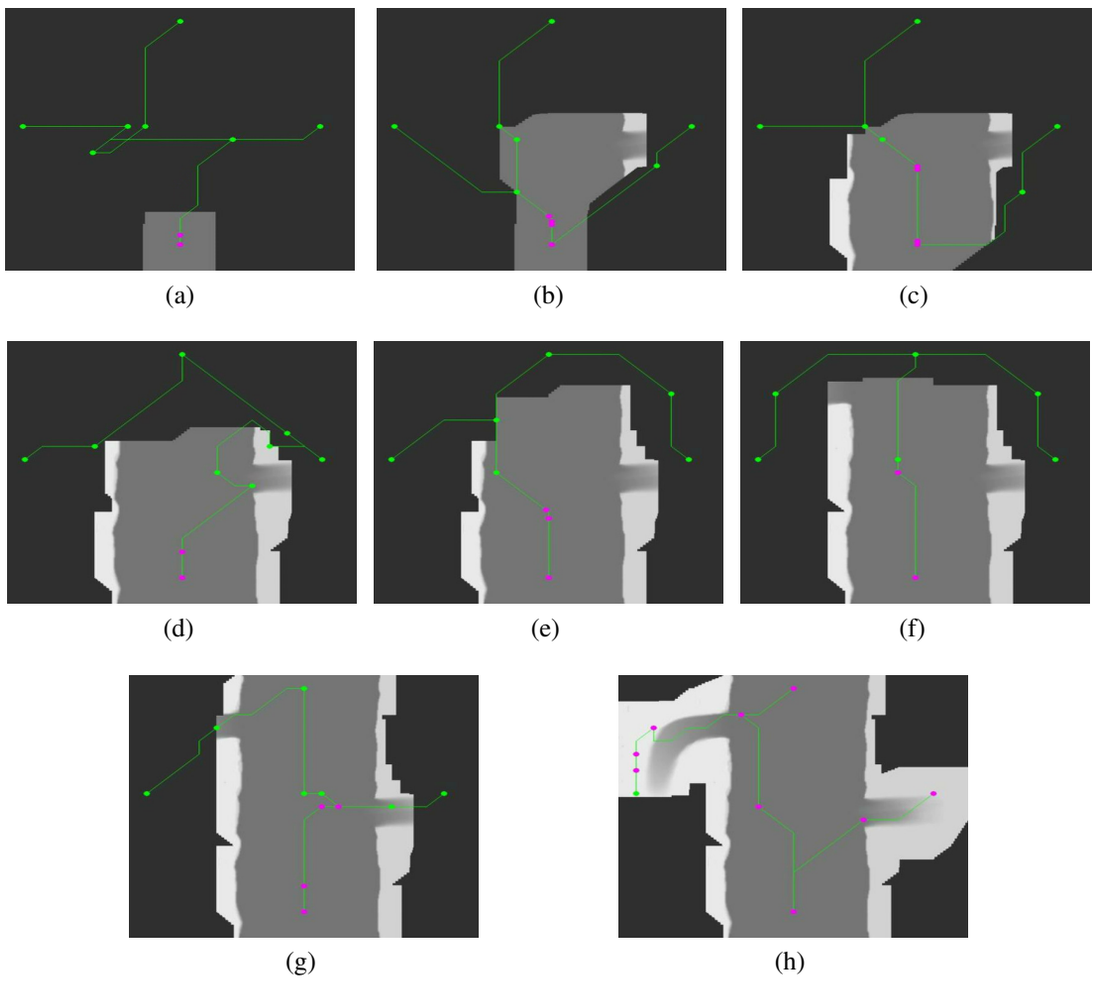


Abbildung 6.7: Instabilität des Planes in unbekannten Umgebungen. Während der Ausführung des Planes ändert sich dieser ständig. Am Ende hat der ursprüngliche Plan nur noch wenig mit dem resultierenden Plan gemeinsam. Besonders problematisch für die Laufzeit ist, dass die Änderungen nicht fließend sind, sondern zwei aufeinander folgende Pläne stark unterschiedlich sein können. Erst wenn ein Großteil der Umgebung erkundet ist, löst der AgentPlaner die gestellte Aufgabe schnell.

Um die Pläne zu stabilisieren und damit die Ausführungsgeschwindigkeit zu erhöhen, wurden verschiedene Modifikationen am AgentPlaner getestet.

Stabilisierung der Wege durch Kostenerhöhung in unbekannten Umgebungen

Ein Grund für die häufigen Neuplanungen ist, dass unbekannte Gebiete grundsätzlich als passierbar gelten. Im Beispiel aus Abbildung 6.6 sieht man ein alternierendes Verhalten. Mit der fortschreitenden Erkundung der Umgebung werden mehr und mehr Teile des Grabens zwischen Start und Zielpunkt sichtbar. Da jedoch die unbekannte Fläche als flach und ohne Hindernis angenommen wird, geht der Algorithmus davon aus, dass er an der unbekannten Fläche den Graben überwinden kann. Dies führt zu einem „Springen" des geplantes Weges, bei dem mal der Weg oben herum gewählt wird und dann wieder unten herum (siehe Abbildung 6.8). Diese Art von ständiger Planänderung führt zu einer deutlich erhöhten Ausführungszeit. Daher muss ein Verfahren implementiert werden, dass solche alternierenden Pläne vermeidet. Dabei wird auf die bekannte Methode, unbekannte Strecken in Pfaden zu bestrafen, zurückgegriffen.
In der ursprünglichen Version des AgentPlaner ist die Streckenlänge mit den Kosten dieser Strecke gleichgesetzt. Dadurch wird zwischen zwei Punkten immer der (euklidisch) kürzeste Pfad gefunden. Diese Tatsache erzeugt die alternierenden Pläne. Sobald z.B. an der unteren Seite ein Teil des Grabens sichtbar wird, ist der Weg oben herum etwas kürzer. Sobald hier jedoch wieder ein Teil des Grabens entdeckt wird, ändert sich dies und der untere Teil erscheint kürzer. Somit muss bei der Pfadplanung auch die Ungewissheit, ob ein Pfad durch unbekanntes Gebiet sinnvoll ist, in die Kosten des Pfades eingerechnet werden. Hier können unterschiedliche Kosten für den unbekannten Bereich angesetzt werden. Explizit werden in dieser Implementierung die Distanzkosten in unbekannten Bereichen verdoppelt. Dies bedeutet, ein bekannter Pfad muss mehr als doppelt so lang sein wie ein unbekannter Pfad, damit der unbekannte Pfad bevorzugt wird. Das bedeutet aber auch, dass ein einmal eingeschlagener Pfad deutlich länger verfolgt wird. Damit ist es möglich, dass ein nicht-optimaler Pfad verfolgt wird. Allerdings wird dieser Nachteil durch die Eliminierung der alternierenden Pläne und die dadurch eingesparte Zeit meist deutlich übertroffen. Im Beispiel des alternierenden Pfades am Graben (Abbildung 6.8) etwa kann die Ausführungzeit für diese Teilsequenz auf die Hälfte gesenkt werden.

Reduzierung der Anzahl der Neuplanungen

Der AgentPlaner ist grundsätzlich in der Lage, auch in unbekannte Umgebungen zu planen. Jedoch benötigen die Roboter durch die häufigen Neuplanungen sehr lange, bis sie ihre Zielpunkte erreichen. Dies liegt vor allem daran, dass der Übergang von den Handlungsanweisungen des alten Planes zu den Handlungsanweisungen des neuen Planes viel Zeit in Anspruch nimmt (siehe Kapitel 6.3).

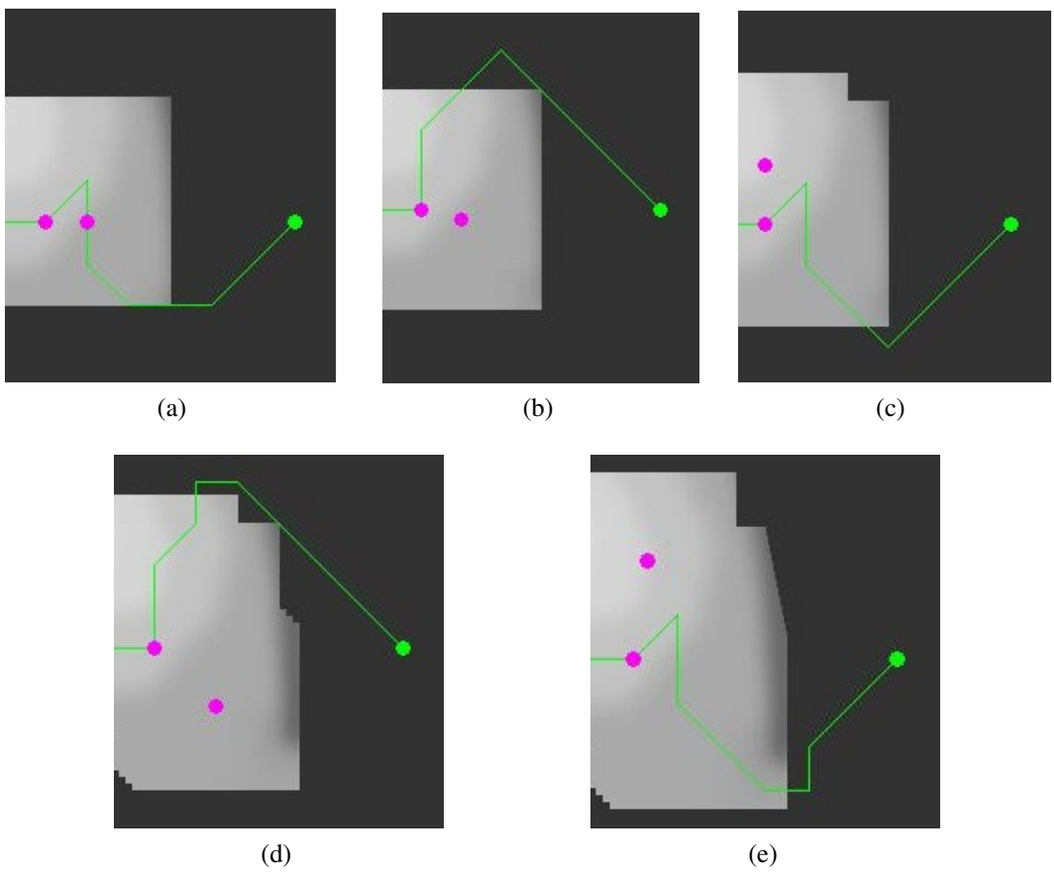


Abbildung 6.8: Alternieren des Planes. Hierbei ist jeweils ein Teil des Planes zu erkennen. Punkte in Magenta kennzeichnen Roboter, die grüne Linie ist der zu nehmende Pfad und grüne Punkte kennzeichnen Zielpositionen. Der Roboter hat einen kleinen Teil des Grabens entdeckt. Zunächst scheint der Weg unten herum der kürzere. Nach einigen Schritten erscheint jedoch der obere Pfad kürzer. Der Roboter ist somit nicht mehr auf dem geplanten Pfad und muss zunächst zurückkehren. Dieses Springen zwischen zwei annähernd gleich guten Pfaden vollzieht sich einige Male. Zuletzt wird ein Pfad über die Brücke gefunden, der auch bestehen bleibt.

Um diesen Effekt abzumildern, wurde versucht, die Pläne zu stabilisieren, d.h. die Anzahl der notwendigen Änderungen zu verkleinern. Ein einfacher Ansatz ist es, die Relaispunkte, auf denen schon ein Roboter steht, in den neuen Plänen weiter zu nutzen. Dies wird dadurch erreicht, dass diese Relaispunkte zu den Terminalen hinzugezählt werden, d.h. diese Punkte können im Optimierungsschritt nicht mehr gelöscht werden. Diese einfache Optimierung führt zwar zu einer schnelleren Ausführungsgeschwindigkeit, aber letztendlich nicht zu dem gewünschten Ergebnis. Es zeigt sich in der Praxis, dass in bestimmten Fällen ein besetzter Relaispunkt R_i nach einer Planänderung trotzdem zurückgezogen werden muss. Zudem erhöht sich die Anzahl benötigter Roboter durch diese Änderung deutlich, da jeder erreichte Relaispunkt in der Endpositionierung verbleibt. Dabei entstehen solche Lösungen wie in Abbildung 6.9.

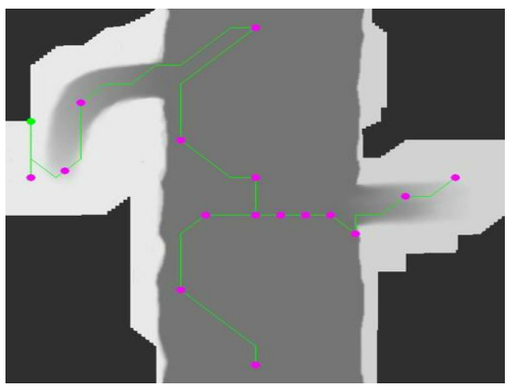


Abbildung 6.9: Lösung des AgentPlaner unter Beibehaltung aller erreichten Relaispunkte. Zwar wird die Ausführungsgeschwindigkeit erhöht, jedoch werden unverhältnismäßig viele Roboter benötigt.

Eine weitere Möglichkeit, die Pläne zu stabilisieren, ist die Relaxierung der Übergänge in unbekannte Gebiete. Bisher ist hier eine konservative Schätzung genutzt worden, ob die Nebenbedingung erfüllt wird. Wenn hier eine optimistische Annahme getroffen wird, ist eine häufige Umplanung möglicherweise nicht mehr nötig. Im relaxierten Fall wird angenommen, dass $C(a, b) = 1$ gilt, sobald auch nur ein Teil der Strecke zwischen a und b unbekannt ist. Insbesondere ist die Nebenbedingung erfüllt, wenn a oder b in unbekannter Umgebung liegen. Der Einfluss auf die Stabilität der Planungen kann in Tabelle 6.1 abgelesen werden. Da die Stabilität der Pläne gleichbedeutend mit der Anzahl der Ausführungsschritte ist, wurden diese gemessen und auf die Anzahl der Planungsschritte des ursprünglichen AgentPlaner normiert. Man kann erkennen, dass die Stabilisierungsmaßnahmen durchaus einen Effekt haben, der die Ausführung des Planes beschleunigt. Tabelle 6.1 zeigt, dass bei der einfachen Distanznebenbedingung die stabilisierenden Maßnahmen die Ausführungszeit negativ beeinflussen. Dies liegt daran, dass bei der Distanz-Nebenbedingung generell nur wenig neu geplant werden muss, da sich der Bedingungsgraph bei der Erkundung nicht ändert. So treten die negativen Eingenschaf-

	Ursprünglich	Beibehaltung Relaispunkte	Bestrafung unbekannter Umgebung	Relaxierte Nebenbedingung
Distanznebenbedingung				
Canyon	1	1,05	0,63	1,33
Landscape	1	3	0,96	0,96
Kommunikationsnebenbedingung				
Canyon	1	0,68	1,47	0,62
Landscape	1	0,47	1,17	0,43
Sichtbarkeitsnebenbedingung				
Canyon	keine Lösung	keine Lösung	keine Lösung	Lösung
Landscape	1	0,59	1,84	0,28

Tabelle 6.1: Einfluss von Plan-stabilisierenden Maßnahmen auf die Ausführungszeit. Die Ausführungszeit des ursprünglichen Ansatzes ist dabei als Maßstab genutzt worden.

ten der Maßnahmen deutlich hervor. Bei den anderen beiden getesteten Nebenbedingungen ist dagegen der Einfluss der stabilisierenden Maßnahmen in einer reduzierten Ausführungsgeschwindigkeit zu erkennen:

- Es zeigt sich, dass eine reine Bestrafung der unbekannten Umgebung bei der Pfadplanung nicht den erhofften Effekt hat. Innerhalb der Distanzbedingung zeigt sich ein geringer positiver Effekt, der jedoch in den anderen Nebenbedingungen nicht gefunden werden kann.
- Die Beibehaltung erreichter Relaispunkte reduziert die Ausführungszeit bei den Nebenbedingungen Kommunikation und Sichtbarkeit merklich. Allerdings ist die extrem hohe Anzahl Roboter, die benötigt wird, nicht akzeptabel (z.B. 17 Roboter in der Canyon-Umgebung gegenüber 8 Robotern mit relaxierter Nebenbedingung).
- Sehr erfolgreich zeigt sich dagegen die Relaxierung der Nebenbedingung in unbekannter Umgebung. Die Ausführgeschwindigkeit kann deutlich erhöht werden, ohne dass ein Anstieg der benötigten Roboter beobachtet wird.

Eine Besonderheit zeigt sich in der Sichtbarkeitsnebenbedingung in der Canyon-Umgebung. Hierbei können mit den oben genannten Annahmen sowohl der ursprüngliche AgentPlaner als auch die Methoden „Beibehaltung erreichter Relaispunkte“ und „Bestrafung unbekannter Umgebung“ keine Lösung finden. Dies liegt daran, dass für eine Lösung die Rampen zu den Plateaus des Canyons befahren werden müssen. Wird aber nun angenommen, dass bei der Berechnung der Sichtbarkeitsnebenbedingung $C_{vis}(a, b)$ bei bekanntem a und unbekanntem b die Punkte die gleiche Höhe haben, so ist die Nebenbedingung an solchen Rampen unerfüllbar. Daher kann kein Weg auf eine unbekannte Rampe gefunden werden. In den beiden anderen Nebenbedingungen wirkt sich dieser

Effekt nicht so gravierend aus. Die Methode „relaxierte Nebenbedingung“ umgeht dieses Problem, da hier $C_{vis}(a, b)$ gleich 1 ist, sobald ein Teil der Strecke zwischen a und b unbekannt ist, also auch wenn b selbst unbekannt ist. Aufgrund dieser Ergebnisse wird die relaxierte Nebenbedingung für die Ausführung in unbekannten Umgebungen vorgeschlagen, wenn nicht ein expliziter Explorationsalgorithmus verwendet werden soll.

6.3 Zusammenführung von Handlungsanweisungen

Ein weiterer, wichtiger Punkt bei der Behandlung von dynamischen und unbekannten Umgebungen ist die Überführung von alten Plänen in neue Pläne. Dabei müssen die ursprünglichen Handlungsanweisungen verworfen, der Übergang geplant und die neuen Handlungsanweisungen für die Roboter berechnet werden. Zudem muss auch hier jederzeit die Nebenbedingung eingehalten werden. Um dieses Problem zu lösen, sind zwei Schritte erforderlich:

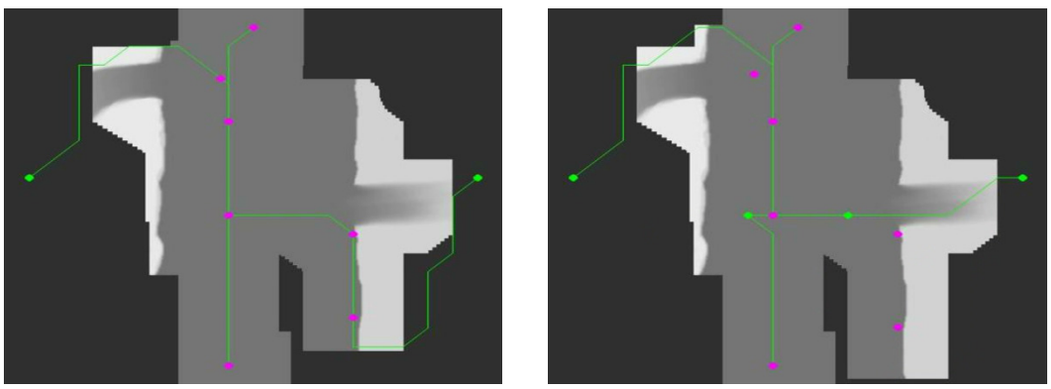
1. Zuführen der Roboter auf den neuen Plan: Durch die Neuplanung ist es wahrscheinlich, dass ein Teil der Roboter nicht (mehr) so positioniert ist, wie es der neue Plan erfordert. Um den aktuellen Plan auszuführen, müssen die Roboter, ohne die Nebenbedingung zu verletzten, auf die nun gültigen Pfade gebracht werden.
2. Handlungsanweisungen von den derzeitigen Positionen aus erstellen: Da die Roboter nach Schritt 1 zwar auf dem aktuellen Plan stehen, jedoch im Allgemeinen nicht auf dem Startpunkt, müssen hierfür passende Handlungsanweisungen erstellt werden.

Dabei ist es weiterhin ein Ziel, den Übergang von einem Plan auf einen anderen so effizient wie möglich zu gestalten.

6.3.1 Zuführen der Roboter auf den neuen Plan

Nachdem ein neuer globaler Mehrroboterplan erstellt wurde, ist es wahrscheinlich, dass einige der Roboter nicht mehr auf dem aktuellen Plan stehen (ein Beispiel aus den Simulationen ist in Abbildung 6.10 zu sehen). Diese Roboter müssen auf den aktuellen Plan bewegt werden, ohne die Nebenbedingung zu verletzten. Allerdings müssen auch Roboter, die auf dem aktuellen Plan stehen, darauf überprüft werden, ob sie dort stehen bleiben können. Für diese Aufgabe wird der alte globale Mehrroboterplan G_{old}, der neue Plan G_{new}, die alte Endpositionierung S_{end}^{old} und die neue Endpositionierung S_{end}^{new} benötigt.
Zunächst wird bestimmt, welche Roboter stehen bleiben können und welche Roboter zurückgezogen werden müssen:

- Stehen bleiben können Roboter, wenn sie auf einem Knoten in S_{end}^{new} stehen und jeder Vorgänger in S_{end}^{new} ebenfalls mit einem Roboter besetzt ist



(a) (b)

Abbildung 6.10: (a) Vor der Neuplanung, (b) nach der Neuplanung. Roboter stehen nach der Neuplanung nicht mehr auf dem aktuellen Mehrroboterplan. In dem Fall müssen die Roboter auf den neuen Plan bewegt werden und danach weitergeführt werden. Dabei darf jedoch nicht die Nebenbedingung verletzt werden.

- Zurückgezogen wird ein Roboter, wenn er nicht auf einem Knoten in S_{end}^{new} steht oder mindestens einer der Vorgänger in S_{end}^{new} nicht mit einem Roboter besetzt ist.

Ein Beispiel für die verschiedenen Fälle wird in Abbildung 6.11 gezeigt.
Die Roboter, die nicht stehen bleiben können, müssen zurück bewegt werden, ohne dabei die Nebenbedingung zu verletzen. Dazu werden die Roboter auf dem Weg des alten Plans G_{old} zurückgeführt. Es muss bestimmt werden, wie weit die Roboter zurückgeführt werden müssen. Dazu wird für jeden Roboter R_i auf dem alten Plan G_{old} der letzte passierte Knoten aus S_{end}^{old} bestimmt. Sei dieser Knoten v_{pass}. Ist v_{pass} ein Knoten, der von einem Roboter, der stehen bleiben darf, besetzt ist, so muss R_i dorthin zurückfahren. Ist dies nicht der Fall, muss der Knoten v_{back} bestimmt werden, zu dem letztendlich zurückgekehrt werden muss. v_{back} ist dabei der erste Knoten auf dem Pfad von v_{pass} zur Wurzel in S_{end}^{old}, der von einem Roboter besetzt ist, der stehen bleiben darf. Solch ein Knoten existiert auf jeden Fall, da hier zumindest die Wurzel, auf der der Leitstand steht, gefunden wird.
Nun ist für jeden Roboter bekannt, zu welchem Knoten er jeweils zurückkehren muss. Um die Nebenbedingung nicht zu verletzten, müssen die Roboter nun in der richtigen Reihenfolge bewegt werden. Dazu muss bestimmt werden, wie weit der Knoten v_{pass} jedes einzelnen Roboters im Graphen S_{end}^{old} von der Wurzel entfernt ist. Zunächst werden all die Roboter einen Knoten in S_{end}^{old} weiterbewegt, die die größte Entfernung zur Wurzel haben. Deren Entfernung zur Wurzel verringert sich dadurch. Sind sie angekommen, werden wieder alle Roboter, die die gleiche, nun weiteste Entfernung zur Wurzel haben, bewegt. Dies wird so lange durchgeführt, bis alle Roboter an ihren jeweiligen Knoten v_{back} angekommen sind. Damit befinden sich alle Roboter wieder auf G_{new}, ohne die

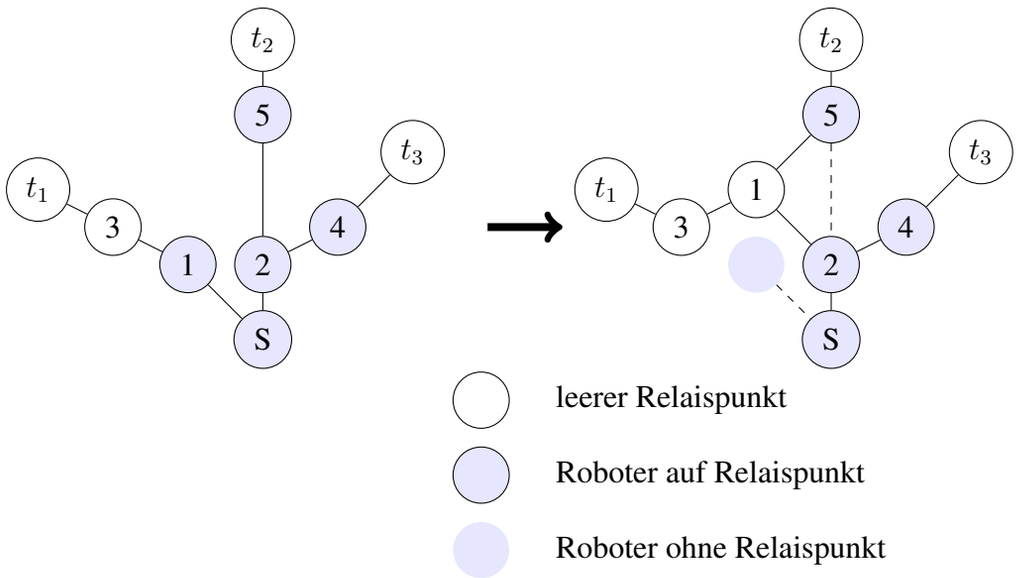
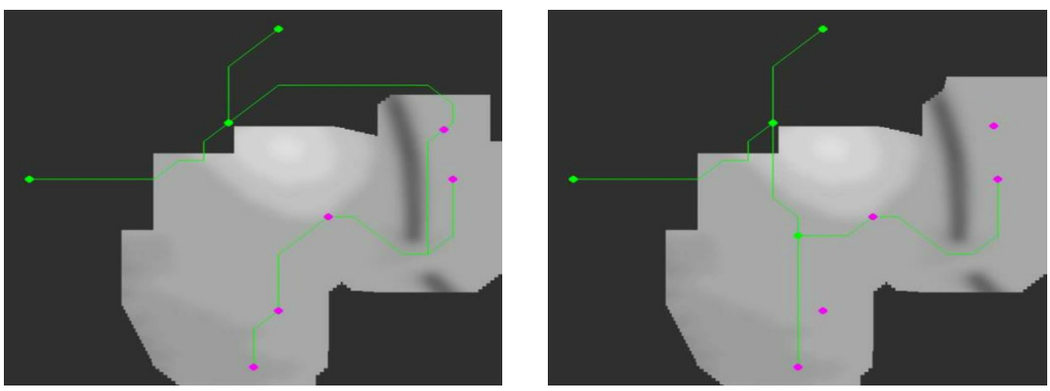


Abbildung 6.11: Schematisches Beispiel für die Bestimmung der Roboter, die stehen bleiben dürfen bzw. die zurückgezogen werden müssen. Links ist der alte globale Mehrroboterplan S_{end}^{old} abgebildet. Dabei haben die Roboter schon die Relaispunkte 1, 2, 4 und 5 erreicht. Durch die weitere Erkundung wird S_{end}^{old} obsolet und S_{end}^{new} (rechte Seite) erzeugt. Um zu bestimmen, welche Roboter zurückgezogen werden, werden alle derzeitigen Roboterpositionen überprüft. Der Roboter ohne Relaispunkt (links neben Relaispunkt 2) muss auf jeden Fall zurückgezogen werden. Der Roboter auf Relaispunkt 2 und der auf Relaispunkt 4 dürfen stehen bleiben, da alle ihre Vorgänger mit Robotern besetzt sind. Der Roboter auf Relaispunkt 5 muss zurückgezogen werden, da auf dem Vorgänger (Relaispunkt 1) kein Roboter steht. Die Roboter, die zurückgezogen werden müssen, nehmen dabei den gestrichelten Pfad.

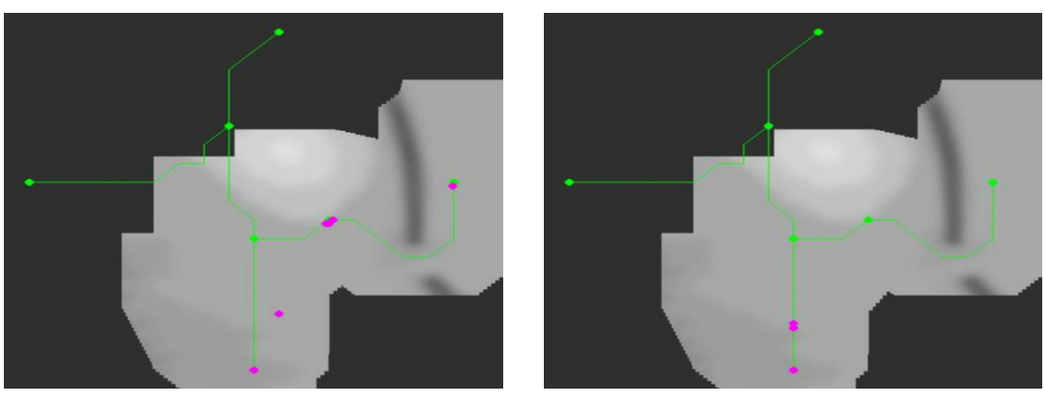
Nebenbedingung verletzt zu haben. Von hier aus können nun die neuen Handlungsanweisungen berechnet werden.

Dabei kann noch eine Optimierung vorgenommen werden. Da es nicht zwangsläufig notwendig ist, die Roboter bis zu einem Relaisknoten zurückzuführen, sondern es ausreicht, sie auf einen gültigen Punkt in P_{new} zurückzuführen, wird der Weg wenn möglich verkürzt. Wenn der Roboter bei der Rückführung nur noch den Knoten v_{back} erreichen muss, wird bei jedem Schritt überprüft, ob der restliche Weg vollständig in P_{new} liegt und kein Relaisknoten mehr dazwischen liegt. Dann kann der Roboter an diesem Punkt stehen bleiben. So kann an bestimmten Stellen die Wegstrecke verringert werden.

Die Rückführung der Roboter auf S_{end}^{new} wird in Abbildung 6.12 dargestellt.



(a) (b)



(c) (d)

Abbildung 6.12: Die verschiedenen Phasen beim Übergang von alten zu neuen Handlungsanweisungen. a) Der alte Plan. Dadurch, dass die Gruppe von Robotern weiter nach oben fährt und ein weiterer Teil des Grabens aufgedeckt wird, wird der Plan ungültig. b) zeigt den neuen Plan. Dabei befinden sich einige Roboter nicht mehr auf dem Plan. In c) sind die Roboter von oben als weitest entfernte Roboter zurückgezogen worden. Zu beachten ist, dass alle Roboter zurückgezogen werden müssen, da der Relaispunkt direkt nach dem Startpunkt nicht besetzt ist. Die Phase des Zurückholens endet, wenn alle Roboter wieder auf dem neuen Plan stehen (siehe d)).

6.3.2 Handlungsanweisungen von den derzeitigen Positionen aus erstellen

Zusätzlich zum neuen Plan G_{new}, müssen neue Handlungsanweisungen erzeugt werden, die die Roboter anhand des neuen Planes steuern. Dazu wird zuerst ermittelt, wie viele Roboter für den neuen Plan benötigt werden. Da hier nur von Lösungen des Agent-Planer ausgegangen wird, kann ausgeschlossen werden, dass temporäre Relaisroboter benötigt werden. Somit werden so viele Roboter benötigt, wie S_{end}^{new} Knoten enthält. Sind dies mehr Knoten als in S_{end}^{old}, müssen weitere Roboter am Startpunkt bereitgestellt werden. In der Simulation werden daher weitere Roboter am Startpunkt initialisiert und können von dort aus verwendet werden. Sind weniger Roboter nötig, als zur Zeit vorhanden, werden die überzähligen Roboter zurück zur Startposition geschickt. Dabei müssen sie natürlich den Weg aus dem Plan G_{new} benutzen, um die Nebenbedingung nicht zu verletzen.
Die Roboter, die in der vorherigen Phase stehen bleiben konnten, werden auch jetzt nicht bewegt. Allen anderen Robotern muss jeweils ein Zielpunkt zugewiesen werden. Dazu werden die Entfernungen aller Roboter zu allen Relaispunkten berechnet und die Roboter so verteilt, dass der Gesamtweg minimiert wird. Den Robotern werden die für diese Zielpunkte notwendigen Handlungsanweisungen von ihrem derzeitigen Standpunkt aus gegeben. Da sie an jedem Relaisknoten auf den dafür zuständigen Roboter warten, muss hierbei auch nicht auf die korrekte Reihenfolge geachtet werden, da sie durch die Handlungsanweisungen implizit erzeugt wird.

6.4 Zusammenfassung der Ergebnisse der Reaktion auf veränderliche Umgebungen

In diesem Kapitel wurde der AgentPlaner auf seine Eigenschaften hin überprüft, in veränderlichen Umgebungen Pläne zu erzeugen. Dabei wurden zwei Arten von Umgebungen betrachtet: Dynamische Umgebungen, die zwar bekannt sind, sich jedoch über die Zeit verändern können, und vollständig unbekannte Umgebungen, die erst erkundet werden müssen.
Aufgrund der Konstruktion der Abbruchbedingung erkennt der AgentPlaner sofort, wenn die vorgeschlagene Endpositionierung nicht mehr gültig ist. Wird nun die Abbruchbedingung nicht als Abbruch, sondern nur als Stopp für die Expansion der Agenten angesehen, so erhält man den AgentPlaner für dynamische Umgebungen. Sobald die Agenten wieder expandieren, finden sie eine neue Endpositionierung. Da dabei schon benutzte Agenten wiederverwendet werden, ändert sich meistens nur der Bereich des Plans, der durch das neue Hindernis beeinflusst wurde.
An Beispielen konnte gezeigt werden, dass der AgentPlaner auch in der Lage, ist auch Pläne für unbekannte Umgebungen zu erzeugen und während der Ausführung diese auch

aktuell zu halten. Dabei wurde der ursprüngliche Algorithmus um Methoden erweitert, die ein häufiges Neuplanen verhindern. Dennoch bilden die Aufgaben der Planungsausführung und der Exploration in unbekannten Umgebungen zwei gegensätzliche Ziele, die nicht einfach zu vereinbaren sind. Daher wird der AgentPlaner als gut geeignet für dynamische Umgebungen angesehen; für eine vollständig unbekannte Umgebung sollte zunächst ein Explorationsalgorithmus eingesetzt werden.
Innerhalb diesen Kapitels wurden nur exemplarisch einige Situationen in dynamischen und unbekannten Umgebungen betrachtet. Dabei konnten gewisse Verhaltensweisen des AgentPlaners aufgezeigt werden. Um die Übertragbarkeit der Ergebnisse z.B. auf beliebige Umgebungen zu zeigen, sollten in Anschlußarbeiten insbesondere systematisch der Einfluss unterschiedlicher Umgebungen betrachtet werden. Hier zeigt die Canyon-Umgebung schon, dass unterschiedliche Klassen von Schwierigkeiten in den Umgebungen zu erwarten sind. In der dynamischen Variante kann zudem weiterführend untersucht werden, in wie weit die Änderung der globalen Mehrroboterpläne durch Entfernen einer Kante, die Ausführungsgeschwindigkeit beeinflusst wird. In den unbekannten Umgebungen ist die Rückführung der Roboter von einem obsoleten Plan auf einen neuen Plan der Teil, der die meiste Zeit in Anspruch nimmt. In den dynamischen, bekannten Umgebungen ist zu Erwarten, dass diese Rückführung nur geringe Zeit in Anspruch nimmt, da Änderungen in Plan nur lokal begrenzt sind.
Abschließend könnte in weiterführenden Arbeiten die Leistungsfähigkeit des AgentPlaner auf unbekannten Umgebungen noch gesteigert werden. Aufgrund der Möglichkeit viele unterschiedliche Nebenbedingungen zu unterstützen, bietet der AgentPlaner gegenüber anderen, in der Literatur bekannten Verfahren, einen Vorteil. Diese Verfahren sind üblicherweise auf eine bestimmte Nebenbedingung zugeschnitten und damit nicht flexible. Um den Vorteil des AgentPlaners auch in unbekannten Umgebungen nutzen zu können, könnte z.B. die Einführung einer Utility-Funktion zur Unterstützung der Exploration erwogen werden.

7 Übertragung auf reale Systeme

Erst mit der Übertragung auf reale Systeme kann gezeigt werden, dass das vorgestellte Planungsframework auch in Applikationen genutzt werden kann. So wurden bei der Entwicklung des STPlaner und AgentPlaner einige Annahmen über die Umgebung gemacht, die in realen Umgebungen nur bedingt haltbar sind. In diesem Kapitel werden die Modifikationen beschrieben, die notwendig sind, um das Planungsframework auf einem realen Mehrrobotersystem zu nutzen. Zudem werden zwei beispielhafte Versuche mit mehreren Robotern beschrieben. Es zeigt sich, dass durch den hohen Abstraktionslevel des Planungsverfahrens viele Probleme der realen Umgebung durch die darunter liegenden ausführenden Schichten aufgefangen werden können. Durch unterschiedliche Hindernisse wird die Einhaltung der Nebenbedingung erschwert. Es kann dennoch gezeigt werden, dass die erzeugten Pläne die Nebenbedingung berücksichtigen und weitgehend einhalten.

7.1 Anpassung für die Realität

Alle bisherigen Ergebnisse wurden innerhalb von Simulationen gezeigt. Damit konnten verschiedene Eigenschaften des Problems und der vorgeschlagenen Algorithmen überprüft werden. Dennoch ist ein Ziel dieser Arbeit, die Verwendbarkeit der vorgestellten Algorithmen in der Praxis zu demonstrieren. Bestimmte Annahmen gelten dabei nur noch eingeschränkt oder gar nicht mehr. Insbesondere die Vielfältigkeit sowie die Kontrollierbarkeit realer Umgebungen werden die in der Simulation gemessenen Ergebnisse verändern.
Zudem sind in der Simulationen artifizielle Karten genutzt worden. Einzige Ausnahme

ist hierbei die Evaluation der Algorithmen in Kapitel 5. Solche artifiziellen Karten sind in weiten Teilen homogen und haben daher stetige Oberflächen. Die Karten, die in diesem Kapitel genutzt werden, sind durch Aufnahmen von 3D-Laserscannern entstanden. Damit sind sie nicht nur mit einem Sensorrauschen belastet, auch die unstrukturierte reale Umgebung sorgt für gewisse Unstetigkeiten. In Unterkapitel 7.3.2 wird die Erstellung der Karte genauer beschrieben. So musste Filter eingesetzt werden, um eine stetige Oberfläche zu erhalten, die notwendig ist, um einen Bewegungsgraphen zu definieren. Dieser Filter reduziert jedoch viele feine Strukturen der Umgebung. Damit ist also zu erwarten, dass die Entscheidung, ob zwischen zwei Knoten eine Verbindung existiert, nicht mehr eindeutig beantwortet werden kann.

Ebenfalls können durch die Filterung Ungenauigkeiten bei der Erstellung des Bedingungsgraphen entstehen. Feine Strukturen können innerhalb der Umgebung verschwinden und daher die Nebenbedingung nicht mehr beeinflussen. Daher ist zu erwarten, dass gerade in den Randbereichen der Nebenbedingungen Ungenauigkeiten auftreten. So wird z.B. das Wellenausbreitungsmodell eine etwas höherer Reichweite voraussagen, da in den digitalisierten Karten die feinen Strukturen nicht mehr vorhanden sind und damit nicht in die Dämpfung des Signals eingehen.

Ebenso hat gerade die Unstetigkeit der Oberfläche Einfluss auf z.B. die Sichtbarkeitsbedingung. Ein Beispiel aus einem realen Testlauf ist in Abbildung 7.1 zu sehen. Damit ist der Bereich der erfüllten Nebenbedingung von einem Punkt aus gesehen deutlich fragmentierter. Dies wird sich durch eine höhere Anzahl von benötigten Robotern in den Plänen widerspiegeln.

Insgesamt zeigt sich, dass die Güte der verfügbaren Karten über die Güte der entstehenden Pläne entscheidet. Insbesondere ist dies für den Steinerbaum-Planer entscheidend, da eine sehr unstetige Umgebung zu Plänen führt, die eine Vielzahl von temporären Relais-Robotern erfordern. Zwar wird in jedem Szenario auch der Plan des STPlaner gezeigt und besprochen, es zeigt sich jedoch, dass die Pläne aus dem AgentPlaner für reale Umsetzungen besser geeignet sind.

7.2 Beschreibung des genutzten Mehrrobotersystems

Für die anschließend beschriebenen Versuche mit einem realen Mehrrobotersystem wurden die Outdoor-Roboter der Forschungsgruppe Unbemannte Systeme des Fraunhofer-Instituts für Kommunikation, Informationstechnik und Ergonomie (FKIE) genutzt. In diesem Unterkapitel werden die verwendeten Roboter mit ihren Fähigkeiten kurz vorgestellt. Zusätzlich werden einige Erklärungen zur verwendeten Softwarearchitektur gegeben.

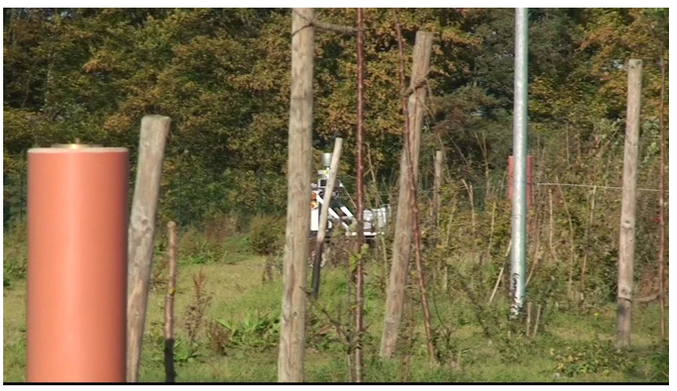


Abbildung 7.1: Beispiel für Probleme, die durch die Diskretisierung des Bewegungs- und Bedingungsgraphen entstehen. Ist hier die Sichtbarkeitsbedingung noch erfüllt? Der Planer nimmt dies an, da der behindernde Baum zu klein ist, um in der Planung berücksichtigt zu werden. Dennoch ist die Sichtverbindung zum Roboter zumindest eingeschränkt.

7.2.1 Hardware

Innerhalb der Versuche kamen unterschiedliche Roboter zum Einsatz. Die Unterschiede beziehen sich hier vor allem auf die Sensorausstattung und die maximale Höchstgeschwindigkeit. Alle Roboter sind rad- oder kettengetrieben und verfügen über eine ähnliche Steigfähigkeit. So kann für alle Roboter der gleiche Bewegungsgraph angenommen werden. Zum Einsatz kamen:

- iRobot ATRV: zwei ATRV junior von iRobot in stark modifizierter Form. Hierbei ist nur noch das Fahrwerk original; die Aufbauten sind vollständig ersetzt worden. Diese Systeme sind mit je einem 2D-Laserscanner für die Hindernisvermeidung ausgerüstet. Sie sind für unebenes Gelände geeignet, können aber aufgrund der Sensorausstattung negative Hindernisse, wie beispielsweise Gräben, nicht erkennen. Ihr Gewicht liegt bei etwa 90 kg und im gemischten Betrieb können sie etwa 2 1/2 Stunden fahren. Ihre Höchstgeschwindigkeit ist auf 0,8 m/s begrenzt.
- telemax: Der telemax von telerob ist in der ursprünglichen Form ein ferngesteuerter Roboter zur Bombenentschärfung. In der hier genutzten Form ist der Arm zugunsten eines Rechners und Sensorik demontiert worden. Auch der telemax ist mit einem 2D-Laserscanner zur Hindernisvermeidung ausgestattet. Der Roboter ist in der Lage, sein Fahrwerk zu verstellen und Kettenteile hinauf oder herunter zu bewegen („Flipper“). Diese Fähigkeit, die auch Treppensteigen ermöglicht, wird jedoch hier nicht genutzt. Damit ist der telemax von seinen Bewegungsfähigkeiten mit den ATRVs gleichzustellen, kann jedoch mit 1,2 m/s deutlich schneller fahren.

- Teodor: Der Teodor als Vorläufermodell des telemax ist in der ursprünglichen Version ebenfalls ein ferngesteuerter Bombenentschärfungsroboter, jedoch deutlich größer und mit etwa 400 kg auch viel schwerer als der telemax. Auch bei diesem Roboter wurden nachträglich Sensoren zur autonomen Fahrt eingebaut. Mit seiner Höchstgeschwindigkeit von etwa 1 m/s entspricht er in seinen Fahreigenschaften etwa denen der ATRVs. Allerdings ist er nicht rad-, sondern kettengetrieben.
- Longcross: Der QinetiQ Longcross ist der größte eingesetzte Roboter. Während der Versuche waren davon zwei im Einsatz, jeweils einer mit Rad- und einer mit Kettenantrieb. Beide Fahrzeuge sind etwa 400 kg schwer und erreichen Höchstgeschwindigkeiten von etwa 15 km/h ($\tilde{4}$,2 m/s). Während der kettengetriebene Roboter ebenfalls nur mit einem 2D-Laser ausgestattet ist, hat der radgetriebene Longcross einen 3D-LIDAR montiert und ist somit auch in der Lage, in schwierigem Terrain zu manövrieren, da der 3D-Laser dem Roboter die Fähigkeit gibt, auch negative Hindernisse (wie z.B. Gräben) zu erkennen und entsprechend zu reagieren.

7.2.2 Software

Die Steuerung eines Mehrrobotersystems ist sehr komplex und erfordert eine Vielzahl von datenverarbeitenden und steuernden Programmen. Dieses notwendige Roboterprogrammierframework konnte natürlich nicht innerhalb dieser Arbeit entwickelt werden. Vor allem die Besonderheiten, die in einem Mehrrobotersystem mit unsicherer Kommunikation auftreten, müssen durch eine Middelware unterstützt werden. Für die Experimente wurde daher auf das Middlewaresystem RoSe (siehe Tiderko et al. [2008]) zurückgegriffen, welches seit 2006 in der Forschungsgruppe Unbemannte Systeme entwickelt wird. Dabei konnte auf ein großes Portfolio von vorhandenen Programmen zur Hardwareansteuerung, Sensordatenverarbeitung und lokaler Navigationsplanung zurückgegriffen werden. Das entwickelte Planungsframework zur koordinierten Navigation unter Nebenbedingungen ist als RoSe Service konzipiert und fügt sich so nahtlos in das System ein.
Im Folgenden werden die genutzten Softwarebausteine einzeln kurz beschrieben. Wurden sie nicht für diese Arbeit speziell entwickelt, sind die Programme als Teil des RoSe-Framework gekennzeichnet und, wenn verfügbar, mit einer Veröffentlichung des jeweiligen Autors versehen.

- Hardwaretreiber:
  - Treiber Laserdistanzmesser: Hier werden die Daten vom 2D-Laserdistanzmesser zur Verfügung gestellt. Die Bildwiederholrate und der Öffnungswinkel sind vom verwendeten Laser abhängig. Dieses Softwaremodul ist Teil des RoSe Frameworks.
  - Treiber Velodyne: Hier werden die 3D-Laserdaten des Velodyne zur Verfügung gestellt. Dieses Softwaremodul ist Teil des RoSe-Frameworks.

  - Treiber IMU: Dieses Softwaremodul stellt Daten der Inertial Measurement Unit zur Verfügung. Dabei werden Daten wie Beschleunigung, GPS Position und Orientierung fusioniert und zur Positionsschätzung genutzt. Der IMU-Treiber ist ebenfalls Teil des RoSe-Frameworks.
  - Treiber GPS: Dieser Hardwaretreiber stellt die GPS Informationen zur Verfügung (Position, Genauigkeit, Anzahl empfangener Satelliten). Er ist Teil des RoSe Frameworks.
  - Treiber Roboter: In diese Kategorie fallen diverse verschiedene Treiber für die verwendeten Roboter. Grundsätzlich ist die Funktionalität ähnlich; so können dem Robotertreiber Translations- und Rotationsgeschwindigkeiten vorgegeben sowie Odometriedaten ausgelesen werden. In Details unterscheiden sich die Treiber durch die auf dem jeweiligen Roboter vorhandene Hardware. Diese Treiber sind Teil des RoSe Frameworks.

- Lokale Navigation

  - Koller: Kollisionsvermeidung und lokale Navigation mit Hilfe des epansive-spaces tree Algorithmus (EST). Der Koller ist ursprünglich für runde Roboter in einer Indoor-Umgebung entwickelt worden (siehe Hoeller et al. [2007]). Er wird in der Forschungsgruppe aber auch für langsame Outdoor-Fahrzeuge, die nur mit einem 2D-Laser ausgestattet sind, genutzt.
  - Lokale Navigation mit adaptiven Motionpattern: Diese lokale Navigation ist speziell für schnelle Roboter mit 3D-Sensorik konstruiert und wird auf allen Robotern mit einem Velodyne genutzt (siehe Hoeller et al. [2010]).

- Globale Navigation

  - GPSPlaner: Ein Service, der eine Folge von GPS-Punkten abfährt. Durch Parametereinstellung kann bestimmt werden, wie genau der Roboter diesem Pfad folgen soll (siehe Hoeller et al. [2008]).
  - Koordinierte Navigation unter spatialen Nebenbedingungen: Der globale Planer, der in dieser Arbeit vorgestellt wird. Hier gibt der Benutzer die Zielpunkte ein, kann bestimmen, ob der Plan die Roboter gesendet wird und kann verfolgen, wie die Roboter den Plan ausführen. Hier werden auch die Handlungsanweisungen erzeugt und an die einzelnen Roboter gesendet.
  - Ausführung der Handlungsanweisungen: Auf jedem Roboter läuft dieser Service, der für diese Experimente auf realen Robotern entwickelt wurde. Jeder Roboter bekommt genau eine Handlungsanweisung zugewiesen. Diese wird dann ausgeführt. Zusätzlich muss der Roboter speichern, wann welcher andere Roboter an ihm vorbeigefahren ist.

Die beiden zuletzt genannten Softwarebausteine sind im Rahmen dieser Arbeit entstanden.

7.3 Reale Experimente mit einem Mehrrobotersystem

Innerhalb dieser Arbeit wurden verschiedene Versuche mit realen Mehrrobotersystemen durchgeführt. Dabei zeigte sich, dass vor allem die Nebenbedingung, ein Kommunikationsnetzwerk zu erhalten, in realen Versuchen eine Herausforderung darstellt. Dies ist vor allem deshalb so, da die Funkverbindung nur in großen Arealen oder stark bebauten Szenarien leicht abreißt; die Vorhersage des genutzten Wellenausbreitungsmodells ist von entscheidender Bedeutung. Allerdings kann eine sehr konservative Schätzung für eine hohe Wahrscheinlichkeit zur Einhaltung der Nebenbedingung sorgen. Hierdurch wird dem Robotersystem zwar ein Teil seiner Beweglichkeit genommen, der Verbindungsabbruch als schlimmstes Szenario wird aber meistens verhindert.

7.3.1 Ermöglichen der Missionserfüllung

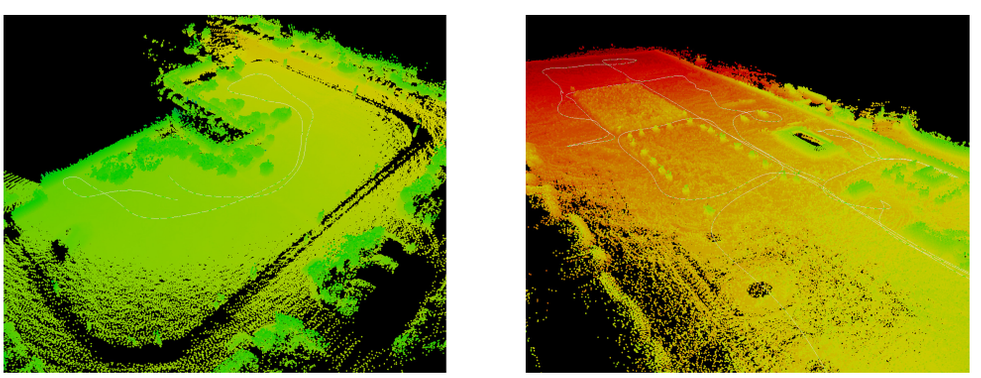
Bei den Tests in den realen Umgebungen wurde vor allem darauf geachtet, dass die gestellte Aufgabe erfüllt werden kann. Diese Aufgabe bestand in erster Linie darin, als einzelner Benutzter eine Gruppe von Robotern zu steuern. Dieser Punkt kann in allen Versuchen als erfüllt betrachtet werden. Auch wenn die durchgeführten Experimente insgesamt einen hohen Personalaufwand erforderten (z.B. um die Roboter in Betrieb zu nehmen oder die Sicherheit zu gewährleisten), ist die Steuerung immer nur von einer Person im Leitstand durchgeführt worden.
Neben der Navigation des MRS gehört zur Missionserfüllung auch die Einhaltung der Nebenbedingung. Dabei wurden zwei verschiedene Szenarien betrachtet. Zum einen soll das MRS zum Aufbau einer Infrastruktur dienen. Dabei sollen die Roboter die Nebenbedingung „Kommunikation“ erfüllen. Dadurch ist sichergestellt, dass vom Leitstand aus zu den Zielpunkten Kommunikation möglich ist. Als zweites Szenario ist die Überwachung von großen Arealen betrachtet worden. Hier ist die Nebenbedingung „Sichtbarkeit“ entscheidend. Da die Roboter sich alle gegenseitig sehen, kann z.B. kein Ereignis, welches sich um das MRS abspielt, übersehen werden. Zudem wäre ein möglicher Ausfall eines Roboter beobachtbar.
Insgesamt konnte in den Experimenten gezeigt werden, dass diese beiden Szenarien gelöst werden können. Der Aufbau von Funkverbindungen funktionierte dabei, gerade um Hindernisse herum, sehr gut. Das Einhalten der Sichtbarkeitsbedingung kann ebenfalls als lösbar bezeichnet werden, allerdings mit Abstrichen, die sich durch Fehler bei der Erstellung des Umgebungsmodells erklären.

7.3.2 Erstellung der Umgebungskarte

Da die vorgestellten Planungsverfahren hauptsächlich für bekannte Umgebungen entwickelt wurden, wird für die beispielhafte Planung in realen Umgebungen eine Karte dieser Umgebung benötigt. Diese Karte wurde in beiden Umgebungen auf die gleiche Weise erzeugt.

Zur Kartenerstellung wurde ein Roboter mit einem Velodyne Laserscanner (siehe Abbildung 7.2) ausgerüstet. Mit diesem 3D-Laserscanner wurde die Umgebung aufgenommen. Die dabei entstehenden Daten sind mit einem 3D-Scan-Matching-Verfahren (siehe Röhling and Schulz [2008]) zu sogenannten Punktwolken der Umgebung zusammengesetzt worden. In Abbildung 7.3 sind Beispiele der Punktwolken zu sehen.

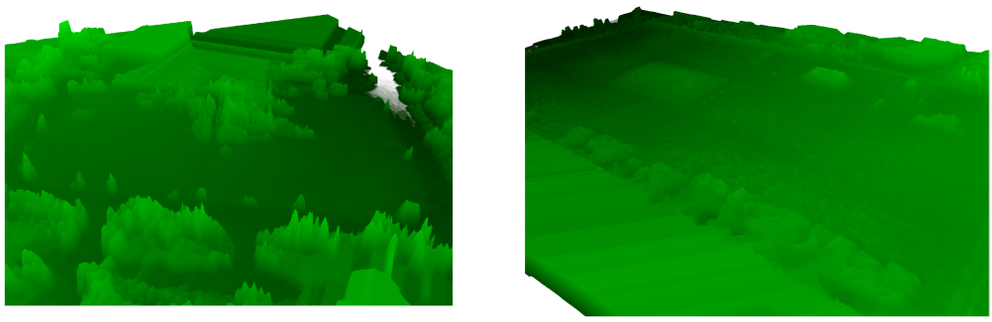
Abbildung 7.2: Roboter der Forschungsgruppe Unbemannte Systeme ausgerüstet mit dem 3D-Laserscanner Velodyne (ganz oben auf dem Roboter).



(a) (b)

Abbildung 7.3: a) Punktwolke des Szenarios: Etablieren einer Funkverbindung (Geb3), b) Punktwolke des Szenarios: Überwachen eines großen Geländes (TDSuS).

Damit das Planungsframework solche Karten nutzen kann, müssen diese in Oberflächen umgewandelt werden, d.h. aus den 3D-Punktwolken müssen sogenannte 2,5-D-Oberflächenmodelle erzeugt werden. Dazu werden zunächst einzeln stehende Punkte gelöscht. Danach wird für eine frei zu wählende Diskretisierung ein Höhengitter erstellt. Es wird die Höhe aller Punkte der Punkwolke ausgewertet, deren (x,y)-Koordinate in das



(a) (b)

Abbildung 7.4: (a) Berechnete Oberfläche der Gebäude3 Karte im Simulationsprogramm. (b) Berechnete Oberfläche der TDSuS Karte im Simulationsprogramm.

entsprechende Feld fallen. Die maximal auftretende Höhe wird dabei als gesamte Höhe des Feldes angenommen. Dabei werden allerdings nicht alle Felder des Höhengitters initialisiert. Damit entsteht eine Oberfläche, die sehr viele große und kleine Löcher enthält. Um die Karte nutzen zu können, müssen diese Löcher gefüllt werden. Es soll also eine einheitliche Oberfläche berechnet werden. Um dies zu erreichen, wird zwischen zwei belegten Zellen linear interpoliert. Um etwaige Artefakte in der Oberfläche zu entfernen, wird zusätzlich noch ein Gauß'scher Weichzeichnungsfilter auf der Oberfläche eingesetzt. Dieser soll vor allem Sensorrauschen reduzieren und den Einfluss von einzelnen Fehlmessungen (Ausreißern) minimieren. In den Simulationen hat sich schon gezeigt, dass eine glatte Oberfläche die Ergebnisse der Planer verbessert. Durch die lineare Interpolation und die Glättung durch den Filter sollen also solch glatte Oberflächen erzeugt werden, ohne die realen Gegebenheiten zu weit zu verfremden. Dabei entstehen dann Karten wie in Abbildung 7.4.

7.4 Optimierung der Planausführung

Innerhalb des hier vorgestellten Planungsverfahrens werden die globalen Mehrroboterpläne und die daraus entstehenden Handlungsanweisungen getrennt betrachtet. Ohne auf die Erzeugung dieser Handlungsanweisungen näher eingehen zu wollen, entsteht durch diese getrennte Betrachtung ein Vorteil bei der Ausführung der globalen Mehrroboterpläne auf einem realen Robotersystem. Dieser Vorteil resultiert aus der Tatsache, dass die Handlungsanweisungen zwischen den Robotern an bestimmten Stellen getauscht werden können. Dies ist nur möglich, wenn die Roboter die gleichen Fähigkeiten habe. Dies ist hier der Fall, so dass eine Optimierung der Planausführung durch den Austausch von Handlungsanweisungen möglich ist.

7.4.1 Austausch von Handlungsanweisungen

Zunächst wird angenommen, dass es keine Handlungsanweisung gibt, die nur von speziellen Robotern ausgeführt werden kann. Soll diese Annahme aufgelöst werden, müssen entsprechend Unterkategorien der Roboter angelegt werden, und Pläne dürfen dann nur innerhalb der Unterkategorien getauscht werden.
Aus einem globalen Mehrroboterplan lässt sich eine Menge von Handlungsanweisungen $\vec{S}$ erzeugen, wobei die Anzahl der Handlungsanweisungen gleich der Anzahl der benötigten Roboter ist. Jede Handlungsanweisung besteht dabei aus einer Sequenz von Handlungen. Die Menge der Handlungsanweisungen wird nun den Robotern $\vec{\mathcal{A}}$ zugeordnet:

$$\vec{S} \rightarrow \vec{\mathcal{A}} \Rightarrow \begin{pmatrix} A \rightarrow \mathcal{A}_1 \\ B \rightarrow \mathcal{A}_2 \\ C \rightarrow \mathcal{A}_3 \\ \vdots \end{pmatrix}$$

In den weiteren Unterkapiteln wird gezeigt, dass mit Hilfe des Tauschoperators die Ausführungszeit der Handlungsanweisungen verringert werden kann. Dabei muss jedoch beachtet werden, dass eine Tauschoperation nicht zu jeder Zeit durchgeführt werden darf. So kann ein Tausch in bestimmten Fälle zur Verletzung der Nebenbedingung führen. Auch wenn der Austausch von Handlungsanweisungen zu anderen Zeitpunkten möglich ist, wird angenommen, dass die Roboter nur Handlungsanweisungen austauschen dürfen, wenn sie so nahe beieinander stehen, dass die Nebenbedingung mit Sicherheit erfüllt ist und kein Hindernis sie trennt (in realen Umgebungen im Bereich von 2-3 Metern Distanz). So ist zum einen gegeben, dass die Roboter miteinander kommunizieren können und so den Plan auch austauschen können. Zum anderen kann so die Nebenbedingung nicht verletzt werden.

7.4.2 Berücksichtigung von unterschiedlichen Robotergeschwindigkeiten

In realen Mehrrobotersystemen sind die Roboter oftmals nicht gleich schnell. Dies kann dazu führen, dass einige Roboter sehr lange an bestimmten Relaispunkten stehen bleiben müssen, z.B. weil dem langsamsten Roboter der erste Relaispunkt zuordnen wird. Damit hat der langsamste Roboter den kürzesten Weg und behindert das Gesamtsystem nur kurz. Allerdings führt dies dazu, dass zunächst alle schnellen Roboter am ersten Relaispunkt warten müssen. Zudem muss die Geschwindigkeit der einzelnen Roboter im Vorhinein bekannt sein, damit die Pläne dementsprechend verteilt werden können. Mit Hilfe des dynamischen Tauschens der Handlungsanweisungen kann dieses Problem automatisiert umgangen werden.
Sei Roboter $\mathcal{A}_i$ der erste Roboter am ersten Relaispunkt R_1. Im globalen Mehrroboterplan sind jedoch noch weitere Relaispunkte vorhanden. Ist für $\mathcal{A}_i$ der Relaispunkt R_1

der endgültige Zielpunkt, so könnten alle weiteren Roboter, die bei R_1 eintreffen, sofort weiterfahren. Allerdings ist anzunehmen, dass $\mathcal{A}_i$ nicht der langsamste Roboter der Gruppe ist, da er R_1 zuerst erreicht hat. Daher wird er, sobald der nächste Roboter $\mathcal{A}_j$ eintrifft, die Handlungsanweisung mit diesem Tauschen. Ist für beide Roboter R_1 nicht der endgültige Zielpunkt, dürften eigentlich beide noch nicht weiterfahren. In diesem Fall wird $\mathcal{A}_j$ temporär eine Handlungsanweisung erzeugen, die R_1 als endgültiges Ziel hat. Damit darf $\mathcal{A}_i$ weiterfahren. So tauscht jeder Roboter, der gerade an R_1 wartet, seine Handlungsanweisung mit dem gerade ankommenden Roboter. Dies gilt nur dann nicht, wenn R_1 das endgültige Ziel des ankommenden Roboters ist. Hier wird nicht getauscht, die eventuell erzeugte temporäre Handlungsanweisung verworfen und die ursprüngliche Handlungsanweisung von dem wartenden Roboter weiterverfolgt.
Dieser Tauschmechanismus hat dabei entscheidende Vorteile:

- Sortierung nach schnellen Robotern: Zwar ist hierdurch nicht garantiert, dass der schnellste Roboter auch den längsten Weg zugeordnet bekommt. Allerdings müssen die langsamsten Roboter nur kurze Wege fahren. Somit wird die Ausführungszeit deutlich verkürzt.
- Kein Vorwissen nötig: Da sich die Roboter über das Tauschsystem anhand der realen Begebenheiten nach Geschwindigkeiten sortieren, ist es nicht nötig, vorher zu wissen, welcher Roboter in welchem Gelände wie schnell ist.
- Kein Stau an den Relaispunkten: Sobald mehr als ein Roboter an einem Relaispunkt angekommen ist, darf mindestens ein Roboter weiterfahren. Dies ist insbesondere bei der realen (nicht simulierten) Ausführung ein deutlicher Vorteil. Eine größere Gruppe eng beieinander stehende Roboter führt bei der Bewegungsplanung üblicherweise zu Problemen und Verzögerungen.

Hier vereinfacht die Trennung von globalen Mehrroboterplänen und lokalen Handlungsanweisungen die Ausführung und führt zu einer selbst-organisierenden, dezentralen Koordinierung der Roboter, bei der die Gruppe die langsamsten Roboter schnell identifiziert und entsprechend behandelt.

7.4.3 Behandlung von Deadlocks und Engstellen

Eine wichtige Funktion bei der Erstellung eines Planes für ein MRS ist die Erkennung und Behandlung von Deadlocks und Engstellen. Beide Effekte treten dann in einem MRS auf, wenn mehrere Roboter gleichzeitig zu den gleichen Zielen fahren oder zusammen auf engem Raum navigieren. Dabei sind Engstellen solche Bereiche, die so eng sind, dass nicht mehr als ein Roboter dort fahren kann. In solchen Bereichen behindern sich Roboter gegenseitig, insbesondere wenn sie unterschiedlich schnell sind oder gegenläufig passieren wollen.
Deadlocks entstehen, wenn zwei oder mehr Roboter wechselseitig darauf warten, dass ein anderer Roboter eine Ressource frei gibt, also z.B. aus dem Weg geht. Solche Deadlocks verlangsamen nicht nur die Ausführung der Handlungsanweisung, sondern führen

dazu, dass der gesamte Plan nicht mehr ausgeführt werden kann. Sie müssen deshalb vermieden werden.
In dem in dieser Arbeit behandelten Problem geht es allein um die Navigation der Roboter. Somit beziehen sich die Probleme, die durch Engstellen oder Deadlocks auftreten können, alleine auf Situationen, die durch die Umgebung verursacht werden. Ein typisches Beispiel für eine Situation, die auch zu einem Deadlock führen kann, ist eine Engstelle, an der zwei Roboter in unterschiedlicher Richtung einander passieren wollen. Solch ein Passieren ohne gegenseitige Behinderung erfordert eine direkte Abstimmung der Roboter und benötigt bei der Ausführung viel Zeit, z.B. weil ein Roboter zunächst zurücksetzten muss. Kann eine solche Situation aber erkannt werden und tauschen die Roboter ihre Handlungsanweisungen, so überwinden quasi die Handlungsanweisungen ohne einen physischen Körper die Engstelle und beide Roboter können nun wegfahren, ohne sich begegnen zu müssen. Der Vorteil dieser Handlungsweise ist, dass nicht schon bei der Erstellung der Handlungsanweisungen darauf geachtet werden muss, dass solche Engstellen vermieden werden. Dies ist auch oft nicht möglich, wenn z.B. die Umgebung nicht bekannt ist oder erst zur Laufzeit die Geschwindigkeiten der Roboter eingeschätzt werden können.
Auch bei Deadlocks aufgrund von Ressourcenkonflikten ist das Tauschen der Handlungsanweisungen eine Möglichkeit, diesen zu lösen. Dabei werden die Handlungsanweisungen, die eine spezielle Ressource erfordern, an den Roboter gegeben, der über diese Ressource verfügt. Dieses Verfahren versagt erst dann, wenn mehr als eine Ressource von einem Roboter benötigt wird und diese Ressourcen so über die restlichen Roboter verteilt sind, dass es keine Handlungsanweisung gibt, für die alle notwendigen Ressourcen bei einem Roboter liegen.

7.5 Versuche

Um die Belastbarkeit der vorgestellten Planungsverfahren auch in der Praxis zu zeigen, sind zwei unterschiedliche Szenarien mit einem realen Robotersystem durchgeführt worden. Das erste Szenario stellt die Aufgabe, eine Funkverbindung über ein Gebäude hinweg zu etablieren. Das zweite Szenario verlangt nach Einhaltung der Sichtbarkeitsbedingung auf einer größeren Fläche, die mit Bäumen und Büschen besetzt ist.

3-Roboter-System zur Etablierung einer Funkverbindung

In diesem Versuch soll das Robotersystem dazu genutzt werden, eine Funkverbindung um ein Gebäude herum zu etablieren. In Abbildung 7.5 ist eine Draufsicht der Umgebung abgebildet. Das Gebäude ist stark gegen elektromagnetische Wellen abgeschirmt, sodass es als undurchlässig für Funkwellen angesehen werden kann. Daher muss, wenn eine Funkverbindung zwischen dem Start- und Endpunkt bestehen soll (siehe Abbildung 7.5), eine Relaisstation an der Seite des Hauses errichtet werden. In Abbildung

7.6a bzw. Abbildung 7.6b sieht man die Lösungen des STPlaner bzw. des AgentPlaner. Durch die Abschirmung des Gebäudes, welches nahezu keinen Funkkontakt zulässt, ähnelt die Kommunikationsnebenbedingung hier der Sichtbarkeitsnebenbedingung. Der Unterschied besteht dabei hauptsächlich darin, dass die Büsche, die um das Gebäude herum angepflanzt sind, die Nebenbedingung nur gering beeinflussen.

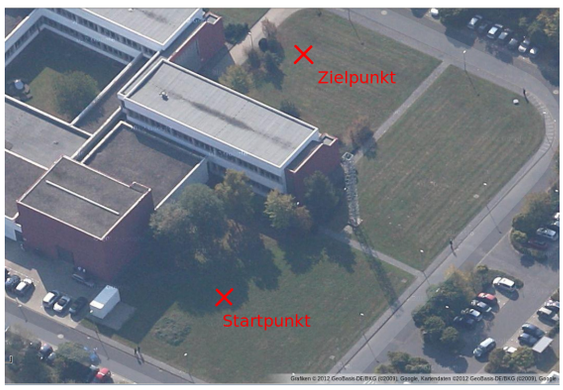
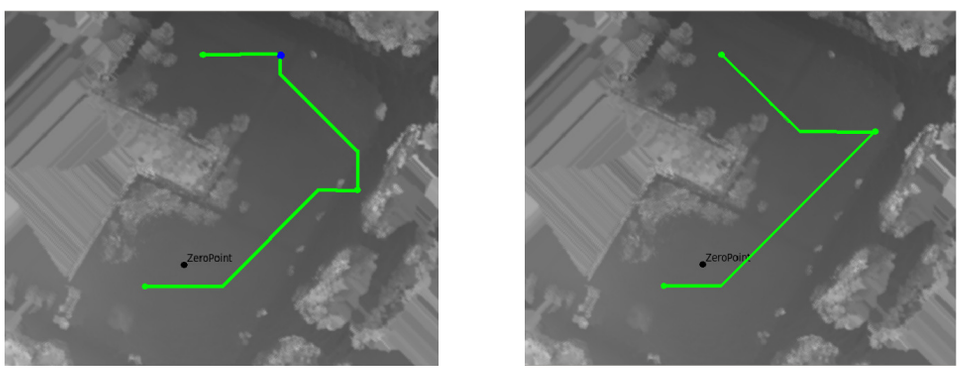


Abbildung 7.5: Gebäude 3 mit Wiesen an drei Seiten. Das Gebäude ist gegen elektromagnetische Wellen abgeschirmt, sodass eine direkte Funkverbindung zwischen Start- und Zielpunkt nicht möglich ist. Über ein Relais soll diese Funkverbindung nun aufgebaut werden. Quelle: Google Maps.



(a) (b)

Abbildung 7.6: (a) Plan des STPlaners. (b) Plan des AgentenPlaners

Beide Planer stellen jeweils einen Roboter als Relaisstation an die Seite des Gebäudes. Bei der Ausführung muss der Roboter, der den Zielpunkt anfahren soll, warten, bis der Relaisroboter angekommen ist.

Die Bilderfolge in Abbildung 7.7 zeigt die Ausführung des Planes, der vom AgentPlaner erzeugt wurde. Hierfür wurden drei Roboter benötigt. Dabei stellt ein Roboter den Leitstand bzw. den Startpunkt dar und ein Roboter wird auf dem Zielpunkt postiert. Bei diesem, genau wie bei allen anderen Testläufen in diesem Szenario, blieb die Funkverbindung zwischen dem Startroboter und dem Roboter für den Zielpunkt jederzeit bestehen. Wurde jedoch der Relaisroboter abgeschaltet, unterbrach dies sofort auch die Verbindung zwischen Start- und Zielroboter.

Startposition des 3-Roboter-Systems

Links steht der Roboter auf der Startposition. Die beiden Roboter im Vordergrund fahren zum ersten Relaispunkt.

Der Roboter rechts hat den Relaispunkt erreicht. Er kann allerdings erst weiterfahren, wenn der andere Roboter auch den Punkt erreicht hat.

Abbildung 7.7: Fotos aus einem Beispiellauf zur Etablierung einer Funkverbindung.

Abbildung 7.8: Luftbild des Robotertestgeländes (TDSuS: 290m x 110m)

6-Roboter-System zur geschützten Überwachung eines Geländes

Bei diesem Szenario ging es sowohl um die Möglichkeit, ein Mehrrobotersystem zur Überwachung eines Geländes zu nutzen als auch in einem einzigen Szenario alle verfügbaren Roboter des FKIE zu nutzen. Dazu wurde ein Überwachungsszenario auf dem Robotertestgelände des FKIE erstellt. Abbildung 7.8 zeigt das Gelände aus der Vogelperspektive. Zur einfachen Nachvollziehbarkeit, ob die Nebenbedingung eingehalten wird, und zur Motivierung des Überwachungsszenarios ist hier die Nebenbedingung „Sichtbarkeit“ gewählt worden. Dabei ist zu beachten, dass die beiden Zielpunkte durch Gebüsch so voneinander getrennt sind, dass zwischen ihnen keine Sichtverbindung besteht. Um das Planungsverfahren zu zwingen, alle sechs Roboter zu nutzen, wurden zwei Flächen definiert, die vom Planer nicht genutzt werden dürfen. Dies sind zwei Hügel auf dem Testgelände. Durch die Wahl der Nebenbedingung „Sichtbarkeit“ würde sonst ein Roboter auf diesem Hügel postiert, der dann das gesamte Gelände einsehen könnte.

Abbildung 7.9 zeigt einen Plan, den der AgentPlaner für dieses Szenario vorschlägt. Hierbei ist der Unterschied zwischen den real gewonnenen Karten und Simulationen deutlich zu erkennen. Bei perfekten Karten kann man davon ausgehen, dass dieses Szenario mit einem einzigen weiteren Relaisroboter gelöst werden kann. Die automatisch gewonnene Karte enthält einige Unebenheiten und die eingetragenen Bäume haben durch den Gaußfilter einen größeren Einflussbereich, als sie es in Wirklichkeit haben. Daher schlägt der AgentPlaner Pläne mit 3 oder 4 permanenten Relais-Robotern vor. Verschiedene Arten von möglichen Plänen sind in Abbildung 7.10 zu sehen.

Der STPlaner benötigt für das gleiche Szenario weniger Roboter für die Endpositionierung (siehe Abbildung 7.11). Dabei findet der STPlaner eine Endkonfiguration in Form eines *Z*, die nur 5 Roboter benötigt. Damit ist diese Lösung um einen Roboter besser als die typischen Lösungen des AgentPlaner. Allerdings werden hier auch temporäre Re-

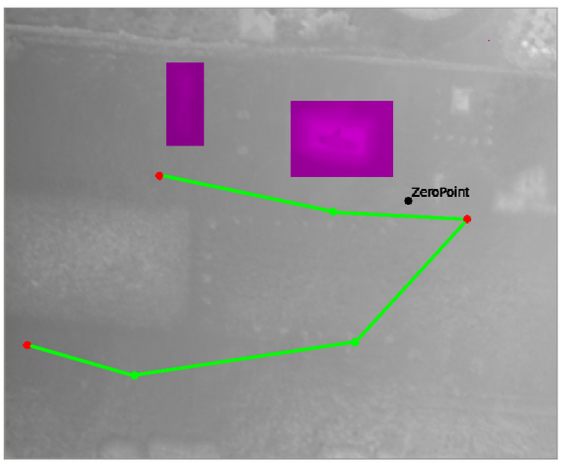
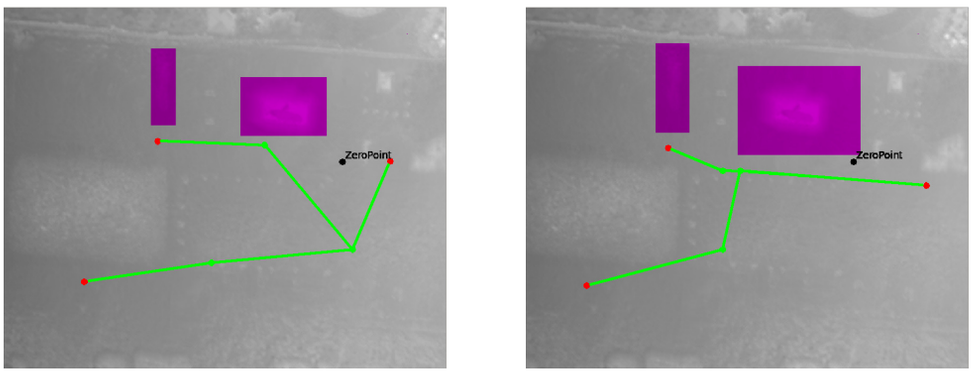


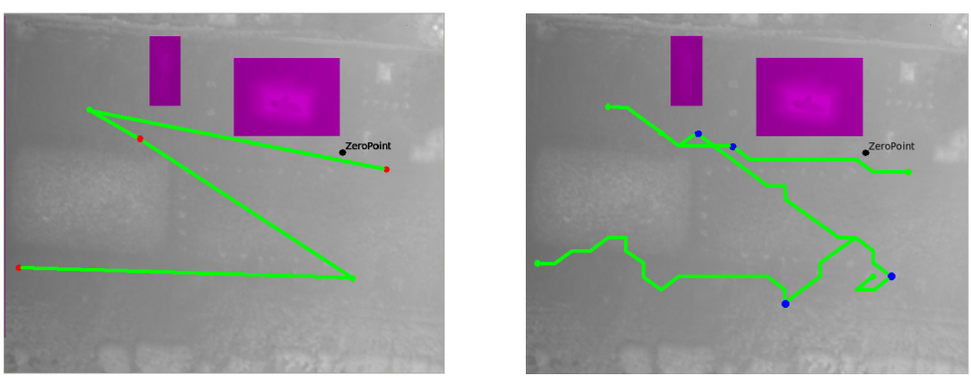
Abbildung 7.9: Ein typischer Plan, vorgeschlagen vom AgentenPlaner für die Aufklärung mit sechs Robotern. Vom Startpunkt aus teilt sich der Weg in zwei Äste auf, die die beiden Zielpunkte mit dem Startpunkt verbinden.

laisroboter benötigt, die für Pläne des AgentPlaner nicht benötigt werden. In Abbildung 7.11b sind die nötigen temporären Relaisroboter in blau eingezeichnet. Hier werden 4 Positionen für TRRs angegeben, was dazu führt, dass effektiv zwei zusätzliche Roboter benötigt werden und damit die Zahl der ausführenden Roboter auf 7 steigt. Daher wurde für das Experiment der AgentPlaner genutzt.
Die Abbildung 7.12 zeigt in einer Bildabfolge eine Testfahrt in diesem Szenario. Bei der Ausführung bleibt die Nebenbedingung „Sichtbarkeit“ beinahe immer erfüllt. Da jedoch einzelne dünne Bäume in der Umgebung verteilt sind und diese bei der gewählten Größe des Bedingungsgraphen nicht ins Gewicht fallen, können kurzzeitig die Sichtverbindungen unterbrochen werden. Solch ein Fall ist z.B. in Abbildung 7.1 zu sehen.



(a) (b)

Abbildung 7.10: Zwei weitere Pläne des AgentPlaner für die Überwachung eines größeren Geländes. Anders als im Plan in Abbildung 7.9 teilen sich die Wege der Roboter in (a) zwar früh, aber erst nach dem ersten Relaispunkt und in (b) erst sehr spät. Grundsätzlich sind die Pläne gleichwertig. Bei der Ausführung kann es bei der späten Teilung des Weges jedoch Probleme geben, da alle Roboter sehr lange denselben Weg und denselben Zielpunkt haben.



(a) (b)

Abbildung 7.11: Plan und zu fahrender Pfad wie er vom STPlaner vorgeschlagen wurde. (a) Die Endkonfiguration des STPlaner benötigt weniger Roboter als die Pläne des AgentPlaner. (b) Jedoch werden viele temporäre Relaisroboter benötigt, um diese Endkonfiguration zu erreichen. Letztendlich benötigt der Plan des STPlaner mehr Roboter als der des AgentPlaners.

Überwachungsaufgabe mit 6 Robotern. Zwei Roboter sollen einen Beobachtungsposten rechts und links eines Waldstückes einnehmen. Dabei soll die Sichtverbindung innerhalb des Mehrrobotersystems nicht unterbrochen werden.

Nach Berechnung des Planes beginnen die Roboter mit dem Ausführen der Handlungsanweisungen. Nach dem Startpunkt und einem Punkt direkt am Container stellt die Laterne den dritten Relaispunkt dar.

Der Telemax (ganz links) hat den dritten Relaispunkt erreicht, damit darf der Longcross (rechts oben), der bis jetzt gewartet hatte, weiterfahren.

Am dritten Relaispunkt trennt sich der globale Mehrroboterplan in zwei Äste auf. Die beiden Longcross (rad- und kettengetrieben) verfolgen die linke Astgabel.

Der Blick vom zweiten Relaispunkt aus, nachdem die Roboter ihre Positionen eingenommen haben. Der telemax (rechts) auf dem dritten Relaispunkt ist deutlich zu erkennen.

Abbildung 7.12: Fotos von einer Ausführung des Plans auf TDSuS.

7.6 Zusammenfassung der Ergebnisse aus der Übertragung auf reale Systeme

In diesem Kapitel wurde die Übertragung des Planungsframeworks zur koordinierten Navigation unter spatialen Nebenbedingungen auf reale Roboter beschrieben. Dabei wurde der Weg von der Simulation zur exemplarischen Einsetzbarkeit gezeigt. Zudem wurden zwei Beispielläufe in realer Umgebung dargestellt.
Zunächst sind einige Unterschiede zwischen den bisher betrachteten Simulationen und der realen Umgebung benannt worden, die zu beachten sind. Danach wurde das System beschrieben, mit dem die Tests durchgeführt wurden. Dazu gehört auch die Integration des Planungsframeworks in das Mehrroboter-Middlewaresystem RoSe. Die Plausibilität des Ansatzes für reale Systeme wurde anhand von zwei Beispielfahrten aufgezeigt.
Diese Beispielfahrten zeigen, dass es einem Operateur möglich ist, mit dem geschaffenen System ein MRS koordiniert zu navigieren. Dabei konnte anhand von zwei Szenarien, dem Aufbau eines Kommunikationsweges sowie der Überwachung eines Geländes, auch gezeigt werden, dass prinzipiell die Nebenbedingungen dazu genutzt werden können, um sinnvolle Aufgaben mit dem MRS zu lösen.
Der Test auf dem realen System zeigt auch, wie wichtig die Koordination der Roboter und die Reihenfolge ihrer Starts ist. Reale Systeme beeinflussen sich gegenseitig. Somit sind Situationen, in denen mehrerer Roboter auf engem Raum aufeinander treffen, zu vermeiden. Hier helfen die Optimierungsmethoden, die durch die Handlungsanweisungen implementiert wurden. Ohne solche Mechanismen, die eine größere Ansammlung von Robotern an einer Position verhindern, steigt die Ausführungszeit erheblich an.
Gerade in größeren Arealen hat sich gezeigt, dass der AgentPlaner die besten, weil leicht ausführbaren Pläne erzeugt. Dieses Ergebnis ist erwartet worden und konnte auch schon in den Simulationen beobachtet werden. Damit ist der Steinerbaum-Planer hauptsächlich als Vergleichsalgorithmus (Baseline) für das Finden einer Endpositionierung wichtig. Der AgentPlaner hingegen hat gezeigt, dass solche Pläne gut und schnell von einem MRS ausführbar sind und im Rahmen der Genauigkeit der Karte auch die gewählte Nebenbedingung in realen Umgebungen einhalten.

8

Diskussion und Ausblick

In dieser Arbeit ist die Problemstellung der koordinierten Navigation eines Mehrrobotersystems unter Nebenbedingungen untersucht worden. Dazu ist zunächst das Problem, welches so in der Literatur noch nicht betrachtet wurde, formal beschrieben worden. Mit der Art der Umgebungsrepräsentation, eine diskretisierte und auf einem Roadmap-Verfahren basierte Graphenstruktur, wurde dann eine spezielle Instanz des Navigationsproblems beschrieben. Das Besondere an dieser Variante ist die Darstellung von Bewegungsfähigkeit und Nebenbedingung in unterschiedlichen Graphen. Dabei bestimmt die Art des Roboters, wie sich der Bewegungsgraph ausgestaltet; die Nebenbedingung definiert den Bedingungsgraph. Diese Repräsentation bildet die Grundlage der weiteren Ergebnisse.

Zunächst wird das Problem der koordinierten Navigation unter Nebenbedingungen in zwei Schritte unterteilt. Der erste Schritt besteht aus dem Finden einer Endpositionierung, d.h. einer Konfiguration für das MRS, die die vom Benutzer gegebenen Zielpunkte enthält, aber auch die Nebenbedingung erfüllt. Der zweite Schritt besteht darin, einen Weg zu der gefundenen Endpositionierung zu bestimmen, der ebenfalls nicht die Nebenbedingung verletzt.

Es konnte gezeigt werde, dass das Finden der Endpositionierung äquivalent zum Finden eines Steinerbaums ist. Damit ist dieses Problem NP-hart. Der Zusammenhang zwischen Steinerbaum und der Endpositionierung wird im Algorithmus STPlaner ausgenutzt. Dieser Algorithmus findet eine Endpositionierung mit Hilfe einer Heuristik für Steinerbäume. Dies hat den Vorteil, dass die Endpositionierung möglichst wenige Roboter benötigt. Zudem sind solche Heuristiken gut untersucht, so dass der STPlaner als „Baseline“ für die Beurteilung weiterer Algorithmen dienen kann. Da der Algorithmus

STPlaner jedoch nicht berücksichtigt, dass die gefundene Endpositionierung von einem Robotersystem angefahren werden soll, und sich somit ineffektive Pläne ergeben, wurde der Algorithmus AgentPlaner entwickelt. Dieser Agenten-basierte Ansatz nutzt sowohl die Information über die Nebenbedingungen als auch die der Bewegungsmöglichkeiten der Roboter aus, um eine Endpositionierung vorzuschlagen. Eine Variante, der Fast-AgentPlaner, wurde aus dem AgentPlaner entwickelt. Dieser kombiniert zwar die Nachteile des STPlaner und des AgentPlaner, benötigt allerdings für das immer noch zufriedenstellende Ergebnis nur einen Bruchteil der Berechnungszeit. Die unterschiedlichen Planungsalgorithmen berechnen die Endpositionierung, aus der sich, zusammen mit dem berechneten Pfad zwischen den Zielpunkten, der sogenannte globale Mehrroboterplan ergibt. Die Frage, wie aus den globalen Mehrroboterplänen Aktionen für den einzelnen Roboter erzeugt werden können, ist in dieser Arbeit nicht thematisiert.

Bei der Entwicklung der Algorithmen wurde auf unterschiedliche Eigenschaften Wert gelegt. So soll der STPlaner Pläne erzeugen, die möglichst wenig permanente Relais-Roboter benötigen, der AgentPlaner hingegen hingegen Pläne, die einfach auszuführen sind und keine temporären Relais-Roboter benötigen. Um diese propagierten Eigenschaften der entwickelten Algorithmen nachzuweisen, wurden ausgiebige Simulationen durchgeführt. Dabei wurden die Algorithmen in unterschiedlichen Umgebungen und unter unterschiedlichen Nebenbedingungen getestet. Die vorhergesagten Eingenschaften der Algorithmen bestätigten sich in diesen Versuchen; daher ist es möglich für bestimmte Szenarien den am besten geeigneten Algorithmus zu wählen.

Für die oben genannten Punkte wurde angenommen, dass die Umwelt statisch ist. Dies ermöglichte, verschiedene Eigenschaften des Navigationsproblems zu zeigen und erste Algorithmen zu entwickeln. Vollständig statische Umgebungen sind jedoch bei einer Nutzung der Ergebnisse auf realen Systemen nicht sinnvoll. Daher wurde der AgentPlaner auf seine Fähigkeiten untersucht, globale Mehrroboterpläne in dynamischen Umgebungen aktuell zu halten. Der AgentPlaner ist dabei in der Lage, sowohl in bekannten Umgebungen auf dynamische (erscheinende) Hindernisse zu reagieren wie auch in vollständig unbekannten Umgebungen zu navigieren. Zusätzlich wurden Optimierungen vorgeschlagen, die die Ausführungszeit in unbekannten Umgebungen verringern.

Zuletzt sind die Ergebnisse in ein reales Robotersystem eingeflossen. Nach der Betrachtung, welche Änderungen der bisherigen Ergebnisse sich durch die Übertragung auf reale Robotersysteme ergeben, und nach der Beschreibung des genutzten Robotersystems wurden exemplarisch zwei reale Versuchsanordnungen beschrieben. Die praktische Umsetzung wurde dabei unterstützt von Optimierungsfunktionen, die die Ausführungsgeschwindigkeit erhöhen.

Ein besonderes Augenmerk wurde in dieser Arbeit auf die Repräsentation der Umgebung und der Nebenbedingungen gelegt. Diese wurden diskretisiert und in unterschiedlichen Graphen darstellt. Dieser Ansatz wurde gewählt, um einen Zugang zu dem neuen Navigationsproblem zu erhalten und die Rechenkomplexität zu begrenzen. Innerhalb der Arbeit konnte, wie oben erwähnt, gezeigt werden, dass die Suche von gültigen Posi-

tionierungen der Roboter dem Steinerbaum-Problem entspricht. Da das Steinerbaum-Problem auch für kontinuierliche Räume bekannt ist und es auch hier Heuristiken gibt, ist es denkbar, das Problem von einer diskreten Struktur auf eine kontinuierliche Umgebung erweitern zu können. Pläne in kontinuierlichem Raum würden dann das Problem vermeiden, dass durch die Diskretisierung mögliche Lösungen verworfen werden. Hier bietet sich eine Möglichkeit, weitere Arbeiten anschließen zu lassen.
Zudem bietet die Trennung der Informationen zu Bewegung und Nebenbedingung weitere Vorteile. So ist es denkbar, dass mehr als ein Nebenbedingungsgraph betrachtet wird und so z.B. abgebildet wird, dass stationäre Relaisknoten in der Kommunikationsnebenbedingung andere Wellenausbreitungscharakteristika haben, als sich bewegende Roboter. Ebenso könnte unterschiedliche Funkhardware abgebildet werden. Genauso kann der Bewegungsgraph durch eine Gruppe von Graphen ersetzt werden, wenn in der Robotergruppe Roboter vorhanden sind, die sehr unterschiedliche Bewegungsfähigkeiten haben (z.B. Boden- und Luftroboter). Bei der Nutzung einer Gruppe von Bewegungs- und Bedingungsgraphen erhöht sich jedoch die Rechenzeit deutlich. Hier ist zu untersuchen, ob die vorgeschlagenen Algorithmen noch sinnvoll genutzt werden können. Dies würde die Flexibilität des vorgestellten Planungsframeworks weiter erhöhen.
Die Abbildung der Nebenbedingung wurde ebenfalls diskretisiert und zu einer binären Entscheidung vereinfacht. Gerade in realen Umgebungen ist eine binäre Entscheidung jedoch schwierig. Es wird ein Übergang von binären Darstellungen zu Wahrscheinlichkeiten vorgeschlagen, die die Nebenbedingungen in realen Umgebungen besser abbilden können. Hier könnte auf Methoden aus der probabilistischen Robotik zurückgegriffen werden.
Zudem wurde in dieser Arbeit die Planung in einen globalen Mehrroboterplan und in Handlungsanweisungen für die Roboter aufgeteilt. Diese Aufteilung bringt den Vorteil, dass Ressourcenkonflikte einfacher gelöst werden können. Daher stellt sich die Frage, ob das Konzept des globalen Mehrroboterplans auf andere Probleme der Mehrroboterplanung angewendet werden kann und ob daraus ein zusätzlicher Nutzen entsteht.
Abschließend ist zu sagen, dass innerhalb dieser Arbeit ein Planungsframework entstanden ist, welches einem einzelne Operateur die Möglichkeit gibt, ein Mehrrobotersystem koordiniert zu bewegen. Dabei sind Erkenntnisse über den Problemkreis gewonnen worden, die in verschiedene Algorithmen umgesetzt wurden. Der Schritt in den Einsatz zur Katastrophenbekämpfung wird jedoch noch eine genauere Repräsentation der Umgebung und den Umgang mit unerwarteten Ereignissen benötigen.

Literaturverzeichnis

Abichandani, P., Benson, H., and Kam, M. (2009). Multi-vehicle path coordination in support of communication. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3237–3244. 24, 26, 29

Alon, N. and Azar, Y. (1992). On-line steiner trees in the euclidean plane. In *SCG '92: Proceedings of the eighth annual symposium on Computational geometry*, pages 337–343, New York, NY, USA. ACM. 34

Arai, T., Pagello, E., and Parker, L. (2002). Guest editorial advances in multirobot systems. *Robotics and Automation, IEEE Transactions on*, 18(5):655–661. 3, 13, 14

Arkin, R. and Diaz, J. (2002). Line-of-sight constrained exploration for reactive multiagent robotic teams. In *7th International Workshop on Advanced Motion Control*, pages 455–461. 22

Arrichiello, F., Das, J., Heidarsson, H., Pereira, A., Chiaverini, S., and Sukhatme, G. (2010). Multi-robot collaboration with range-limited communication: Experiments with two underactuated asvs. In Howard, A., Iagnemma, K., and Kelly, A., editors, *Field and Service Robotics*, volume 62 of *Springer Tracts in Advanced Robotics*, pages 443–453. Springer Berlin Heidelberg. 29

Averbakh, I. and Berman, O. (1996). A heuristic with worst-case analysis for minimax routing of two travelling salesmen on a tree. *Discrete Applied Mathematics*, 68(1-2):17–32. 7

Bahl, P. and Padmanabhan, V. (2000). Radar: an in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775–784. 35, 40

Barnes, D. and Gray, J. (1991). Behaviour synthesis for co-operant mobile robot control. In *International Conference on Control 1991. Control '91*, volume 2, pages 1135–1140. 12

Basu, P. and Redi, J. (2004). Movement control algorithms for realization of fault-tolerant ad hoc robot networks. *IEEE network*, 18(4):36–44. 26

Batalin, M. and Sukhatme, G. (2002). Spreading out: A local approach to multi-robot coverage. In Asama, H., Arai, T., Fukuda, T., and Hasegawa, T., editors, *Distributed Autonomous Robotic Systems 5*, pages 373–382. Springer Japan. 6, 21

Bauer, F. and Varma, A. (1995). Aries: A rearrangeable inexpensive edge-based on-line steiner algorithm. *IEEE Journal of Selected Areas in Communications*, 15:382–397. 33

Bennewitz, M., Burgard, W., and Thrun, S. (2001). Optimizing schedules for prioritized path planning of multi-robot systems. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 271–276. 18

Blin, L., Potop-Butucaru, M., and Rovedakis, S. (2009). A superstabilizing log (n)-approximation algorithm for dynamic steiner trees. *Stabilization, Safety, and Security of Distributed Systems*, pages 133–148. 54, 97

Brüggemann, B., Brunner, M., and Schulz, D. (2013). Asynchronous flooding planner for multi-robot navigation. In *ICINCO 2013 - Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 222–230. 8

Brüggemann, B., Langetepe, E., Lenerz, A., and Schulz, D. (2012). From a multi-robot global plan to single robot actions. In *Proceedings of 9th International Conference on Informatics in Control, Automation and Robotics, ICINCO*, pages 419–422. 8, 78

Brüggemann, B. and Schulz, D. (2010a). Coordinated navigation for multi-robot systems with additional constraints. In van der Hoek, W., Kaminka, G. A., Lespérance, Y., Luck, M., and Sen, S., editors, *AAMAS*, pages 1511–1512. IFAAMAS. 8

Brüggemann, B. and Schulz, D. (2010b). Coordinated navigation of multi-robot systems with binary constraints. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3854–3859. 8

Brüggemann, B., Tiderko, A., and Stilkerieg, M. (2009). Adaptive signal strength prediction based on radio propagation models for improving multi-robot navigation strategies. In *Robot Communication and Coordination, 2009. ROBOCOMM '09. Second International Conference on*, pages 1–6. 40

Brüggemann, B., Brunner, M., and Schulz, D. (2012). Outdoor navigation with a coordinated multi-robot system that maintains spatial constraints. In *Multivehicle Systems*, volume 2, pages 1–6. 8

Brüggemann, B., Brunner, M., and Schulz, D. (2013a). Spatially constrained coordinated navigation for a multi-robot system. *Ad Hoc Networks*, 11(7):1919 – 1930. 8

Brüggemann, B., Langetepe, E., and Lenerz, A. (2013b). A strategic occupation game on graphs. In *Proceedings of Middle-European Conference on Applied Theoretical Computer Science (MATCOS)*. 8, 77

Buckley, S. (1989). Fast motion planning for multiple moving robots. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 322–326 vol.1. 18

Burgard, W., Moors, M., Fox, D., Simmons, R., and Thrun, S. (2000). Collaborative multi-robot exploration. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 1, pages 476–481. 20

Cao, Y., Fukunaga, A., Kahng, A., and Meng, F. (1995). Cooperative mobile robotics: antecedents and directions. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 1, pages 226–234. 12, 13

Carpin, S. and Parker, L. E. (2002). Cooperative leader following in a distributed multi-robot system. In *Proceedings- IEEE International Conference on Robotics and Automation*, volume 3, pages 2994–3001. 19

Chiang, C., Sarrafzadeh, M., and Wong, C. (1990). Global routing based on steiner min-max trees. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 9(12):1318–1325. 50

Clark, C., Rock, S. M., and Latombe, J.-C. (2003). Motion planning for multiple mobile robot systems using dynamic networks. In *IEEE Int. Conference on Robotics and Automation*, pages 4222–4227. 18

Cui, R., Gao, B., and Guo, J. (2012). Pareto-optimal coordination of multiple robots with safety guarantees. *Autonomous Robots*, 32:189–205. 18

De Gennaro, M. C. and Jadbabaie, A. (2006). Decentralized control of connectivity for multi-agent systems. In *Proceedings of the 45th IEEE Conference on Decision & Control*, pages 3628–3633. 26

Ding, S. and Ishii, N. (2000). An online genetic algorithm for dynamic steiner tree problem. In *Industrial Electronics Society, 2000. IECON 2000. 26th Annual Confjerence of the IEEE*, volume 2, pages 812–817. 54, 97

Doniec, A., Bouraqadi, N., Defoort, M., Stinckwich, S., et al. (2009). Distributed constraint reasoning applied to multi-robot exploration. In *Tools with Artificial Intelligence, 2009. ICTAI'09. 21st International Conference on*, pages 159–166. IEEE. 28

Dudek, G., Jenkin, M., Milios, E., and Wilkes, D. (1996). A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375–397. 14, 15

Elmaliach, Y., Agmon, N., and Kaminka, G. (2007). Multi-robot area patrol under frequency constraints. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 385–390. 19

Escoffier, B., Milanič, M., and Paschos, V. (2009). Simple and fast reoptimizations for the steiner tree problem. *Algorithmic Operations Research*, 4(2):86–94. 34

Fierro, R., Das, A., Spletzer, J., Esposito, J., Kumar, V., Ostrowski, J., Pappas, G., Taylor, C., Hur, Y., Alur, R., et al. (2002). A framework and architecture for multi-robot coordination. *The International Journal of Robotics Research*, 21(10-11):977. 17

Fox, D., Burgard, W., Kruppa, H., and Thrun, S. (2000). A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8(3):325–344. 19

Gage, D. (1994). Command control for many-robot systems. *Unmanned Systems*, 10(4):28–34. 6

Gilbert, E. and Pollak, H. (1968). Steiner minimal trees. *SIAM Journal on Applied Mathematics*, pages 1–29. 49

Godsil, C. D. and Royle, G. (2001). *Algebraic graph theory*, volume 8. Springer. 26

Goldhirsh, J. and Vogel, W. J. (1998). Handbook of propagation effects for vehicular and personal mobile satellite systems. *NASA Reference Publication*, 1274. 40, 67

Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107. 38

Hashemi, H. (1993). The indoor radio propagation channel. *Proceedings of the IEEE*, 81(7):943–968. 35

Hoeller, F., Roehling, T., and Koenigs, A. (2008). Combining coordinated navigation and reactive collision avoidance for gps-based convoying. In *In Proc. of Towards Autonomous Robotic Systems*, pages 93–100. 18, 119

Hoeller, F., Röhling, T., and Schulz, D. (2010). Offroad navigation using adaptable motion patterns. In *Proceedings of International Conference on Informatics in Control, automation and Roboics (ICINCO)*, volume 2, pages 186–191. 119

Hoeller, F., Schulz, D., Moors, M., and Schneider, F. (2007). Accompanying persons with a mobile robot using motion prediction and probabilistic roadmaps. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1260–1265. 119

Hong, X., Xue, T., Kuh, E. S., Cheng, C.-K., and Huang, J. (1993). Performance-driven steiner tree algorithm for global routing. In *Proceedings of the 30th international Design Automation Conference*, DAC '93, pages 177–181, New York, NY, USA. ACM. 50

Howard, A., Mataric, M., and Sukhatme, G. S. (2002). An incremental deployment algorithm for mobile robot teams. *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, 3:2849–2854. 6, 21

Howard, A., Matarić, M., and Sukhatme, G. (2003). Localization for mobile robot teams: A distributed mle approach. In Siciliano, B. and Dario, P., editors, *Experimental Robotics VIII*, volume 5 of *Springer Tracts in Advanced Robotics*, pages 146–155. Springer Berlin Heidelberg. 19

Hwang, F. K. and Richards, D. S. (1992). Steiner tree problems. *Networks*, 22(1):55–89. 33

Iocchi, L., Marchetti, L., and Nardi, D. (2011). Multi-robot patrolling with coordinated behaviours in realistic environments. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2796–2801. 19

Iocchi, L., Nardi, D., and Salerno, M. (2001). Reactivity and deliberation: A survey on multi-robot systems. In *Balancing Reactivity and Social Deliberation in Multi-Agent Systems, From RoboCup to Real-World Applications (selected papers from the ECAI 2000 Workshop and additional contributions)*, pages 9–34, London, UK, UK. Springer-Verlag. 16, 17, 157

Jnaneshwar, D., Heidarsson, H., Pereira, A., Chiaverini, S., and Sukhatme, G. S. (2009). Multi-robot collaboration with range-limited communication: Experiments with two underactuated asvs. In *In Proceedings 2009 International Conference on Field and Service Robots*. 29, 41

Kalech, M., Kaminka, G., Meisels, A., and Elmaliach, Y. (2006). Diagnosis of multi-robot coordination failures using distributed csp algorithms. In *Proceedings of the national Conference on Artifical Intelligence*, volume 21, page 970. 18

Kaminka, G. A., Erusalimchik, D., and Kraus, S. (2010). Adaptive multi-robot coordination: A game-theoretic perspective. In *IEEE International Conference on Robotics and Automation*, pages 328–334. 18

Klein, R. (2005). *Algorithmische Geometrie: Grundlagen, Methoden, Anwendungen*. EXamen. press Series. Springer. 31

Koenig, S., Zheng, X., Tovey, C., Borie, R., Kilby, P., Markakis, V., and Keskinocak, P. (2008). Agent coordination with regret clearing. In *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence*, pages 101–107. 20

Kou, L., Markowsky, G., and Berman, L. (1981). A fast algorithm for steiner trees. *Acta informatica*, 15(2):141–145. 32

Kowalczyk, W. (2001). Multi-robot coordination. In *Proc. of the 2nd Internat. Workshop on Robot Motion and Control*, pages 219–223. 19

Kulich, M., Rollo, M., Mázl, R., Chudoba, J., Benda, P., Přeučil, L., Pěchouček, M., and Štěpán, P. (2007). Multi-robot exploration using multi-agent approach. In *Proceedings of the 13th IASTED International Conference on Robotics and Applications*, pages 495–500. 20

Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers. 11, 12, 31

Lim, H. and Kim, C. (2000). Multicast tree construction and flooding in wireless ad hoc networks. In *Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, MSWIM '00, pages 61–68. 49

Liu, S., Sun, D., and Zhu, C. (2010). Motion planning of multirobot formation. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3848–3853. 19

Mataric, M. J. (1994). Interaction and intelligent behavior. Technical report. 12

Mehlhorn, K. (1988). A faster approximation algorithm for the steiner problem in graphs. *Inf. Process. Lett.*, 27(3):125–128. 6, 7, 32, 53

Mosteo, A. R., Montano, L., and Lagoudakis, M. G. (2008). Multi-robot routing under limited communication range. In *IEEE International Conference on Robotics and Automation (ICRA) 2008*, pages 1531–1536. 30, 41, 147

Mosteo, A. R., Montano, L., and Lagoudakis, M. G. (2009). Guaranteed-performance multi-robot routing under limited communication range. In *Distributed Autonomous Robotic Systems 8*, pages 491–502. 30, 147

Mostofi, Y. (2008). Communication-aware motion planning in fading environments. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3169–3174. 23

Mostofi, Y., Malmirchegini, M., and Ghaffarkhah, A. (2010). Estimation of communication signal strength in robotic networks. In *2010 IEEE International Conference on Robotics and Automation*, pages 1946–1951. 40

Mu, H., Bailey, T., Thompson, P., and Durrant-Whyte, H. (2011). Decentralised solutions to the cooperative multi-platform navigation problem. *Aerospace and Electronic Systems, IEEE Transactions on*, 47(2):1433 –1449. 20

Natalizio, E., Caro, G. D., Sekercioglu, A., Yanmaz, E., Rawat, D. B., Heijenk, G., Weigle, M. C., Bista, B. B., and Chen, Y.-S., editors (2013). *Theory, Algorithms and Applications of Wireless Networked Robotics Recent Advances in Vehicular Communications and Networking*, volume 11. 23

Nguyen, H., Everett, H., Manouk, N., and Verma, A. (2002). Autonomous mobile communication relays. In *SPIE Proc. 4715: Unmanned Ground Vehicle Technology IV*, pages 50–57. 19

Nguyen, H., Pezeshkian, N., Gupta, A., and Farrington, N. (2004). Maintaining communication link for a robot operating in a hazardous environment. In *ANS 10th Int. Conf. on Robotics and Remote Systems for Hazardous Environments*, pages 28–31. 19

Nguyen, H., Pezeshkian, N., Raymond, M., Gupta, A., and Spector, J. (2003). Autonomous communication relays for tactical robots. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, pages 35–40. 19

Notarstefano, G., Savla, K., Bullo, F., and Jadbabaie, A. (2006). Maintaining limited-range connectivity among second-order agents. In *American Control Conference, 2006*, page 6 pp. 24

Parker, L. and Howard, A. (2009). Assistive formation maintenance for human-led multi-robot systems. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 2350–2355. 19

Parker, L. E. (2008). Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents*, 2(1):5–14. 17, 20

Parker, L. E., Kannan, B., Tang, F., and Bailey, M. (2004). Tightly-coupled navigation assistance in heterogeneous multi-robot teams. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1016–1022. 19

Pei, Y., Mutka, M. W., and Xi, N. (2010). Coordinated multi-robot real-time exploration with connectivity and bandwidth awareness. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5460–5465. 23, 27, 28

Pezeshkian, N., Nguyen, H., and Burmeister, A. (2006). Unmanned ground vehicle non-line-of-sight operations using relaying radios. *IASTED Robotics and Applications (RA 2006)*, pages 17–20. 19

Pierre Fraigniaud, L. et al. (2006). Collective tree exploration. *Networks*, 48(3):166–177. 7

Premvuti, S. and Yuta, S. (1990). Consideration on the cooperation of multiple autonomous mobile robots. In *Intelligent Robots and Systems '90. 'Towards a New Frontier of Applications', Proceedings. IROS '90. IEEE International Workshop on*, pages 59–63 vol.1. 13

Qayyum, A., Viennot, L., and Laouiti, A. (2002). Multipoint relaying for flooding broadcast messages in mobile wireless networks. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 3866–3875. 7, 36, 54

Rausch, W. and Levi, P. (1996). Asynchronous and synchronous cooperation-demonstrated by deadlock resolution in a distributed robot system. *Proceedings of Distributed Autonomous Robotic Systems*, 2:245–256. 7, 18

Rekleitis, I., Dudek, G., and Milios, E. (1998). On multiagent exploration. *Visual Interface*, pages 455–461. 20

Rice, S. O. (1944). Mathematical analysis of random noise. *Bell Systems Tech. J.*, 23:282–332. 35

Röhling, T. and Schulz, D. (2008). Improving 3D scan registration for SLAM with clustering and deterministic annealing. In *Proceedings of Towards Autonomous Robotic Systems (TAROS)*, pages 50–56. 121

Rooker, M. N. and Birk, A. (2007). Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, 15(4):435 – 445. 6, 25

Schneider, F. E. and Wildermuth, D. (2012). Influences of the robot group size on cooperative multi-robot localisation—analysis and experimental validation. *Robotics and Autonomous Systems*, 60(11):1421 – 1428. 19

Seidel, S. and Rappaport, T. (1992). 914 mhz path loss prediction models for indoor wireless communications in multifloored buildings. *Antennas and Propagation, IEEE Transactions on*, 40(2):207–217. 35

Shehory, O. and Kraus, S. (1998). Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2):165 – 200. 20

Simeon, T., Leroy, S., and Lauumond, J.-P. (2002). Path coordination for multiple mobile robots: a resolution-complete algorithm. *Robotics and Automation, IEEE Transactions on*, 18(1):42–49. 18

Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S., and Younes, H. (2000). Coordination for multi-robot exploration and mapping. In *Proceedings of the National conference on Artificial Intelligence*, pages 852–858. 20

Spanos, D. and Murray, R. (2005). Motion planning with wireless network constraints. In *Proceedings of the American Control Conference*, pages 87–92. 22, 26

Svestka, P. and Overmars, M. H. (1998). Coordinated path planning for multiple robots. *Robotics and Autonomous Systems*, 23:125–152. 18

Tanner, H. and Kumar, A. (2005). Towards decentralization of multi-robot navigation functions. In *Proceedings of IEEE International Conference on the Robotics and Automation (ICRA)*, pages 4132–4137. 19

Tc-REs, E. (1995). Radio equipment and systems (res); high performance radio local area network (hiperlan); functional specification. *ETSI*, 6:921. 7, 35

Tiderko, A., Bachran, T., Hoeller, F., and Schulz, D. (2008). Rose–a framework for multicast communication via unreliable networks in multi-robot systems. *Robotics and Autonomous Systems*, 56(12):1017–1026. 118

Todt, E., Rausch, G., and Suarez, R. (2000). Analysis and classification of multiple robot coordination methods. In *Robotics and Automation, Proceedings. ICRA '00. IEEE International Conference on*, volume 4, pages 3158–3163. 16

Tuna, G., Gulez, K., and Gungor, V. C. (2013). The effects of exploration strategies and communication models on the performance of cooperative exploration. *Ad Hoc Networks*, 11(7):1931 – 1941. 23, 34

Urcola, P. and Montano, L. (2009). Cooperative robot team navigation strategies based on an environment model. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4577–4583. 19

van den Berg, J. and Overmars, M. (2005). Prioritized motion planning for multiple robots. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 430–435. 18

Van Tuan, L., Bouraqadi, N., Stinckwich, S., Moraru, V., and Doniec, A. (2009). Making networked robots connectivity-aware. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3502–3507. 24, 25

Velagapudi, P., Sycara, K., and Scerri, P. (2010). Decentralized prioritized planning in large multirobot teams. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4603–4609. 18

Viennot, L. (1998). Complexity Results on Election of Multipoint Relays in Wireless Networks. Rapport de recherche RR-3584, INRIA. 36

Wagner, A. and Arkin, R. (2004). Multi-robot communication-sensitive reconnaissance. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 5, pages 4674–4681. 147

Waxman, B. (1988). Routing of multipoint connections. *Selected Areas in Communications, IEEE Journal on*, 6(9):1617–1622. 49

Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82. 2

Xu, J., Hong, X., Jing, T., Cai, Y., and Gu, J. (2003). An efficient hierarchical timing-driven steiner tree algorithm for global routing. *Integration, the {VLSI} Journal*, 35(2):69 – 84. 50

A

Dynamische Aufrechterhaltung der Nebenbedingungen

Innerhalb dieser Arbeit wird die koordinierte Navigation unter spatialen Nebenbedingungen mit Hilfe von statischen Positionierungen behandelt. In der hier bearbeiteten Instanz wird dafür ein globaler Mehrroboterplan erstellt, indem eine Endpositionierung berechnet wird. Das bedeutet, für die vorgegebenen Ziele werden Relaispunkte gesucht, damit die Nebenbedingung eingehalten wird. Dies führt unter anderem dazu, dass das Navigationsproblem auf das Steinerbaum-Problem zurückgeführt werden kann. Diese Lösung kann als statisch angesehen werden, da ein Roboter, der seinen Zielpunkt erreicht hat, nicht mehr bewegt wird.

Dem kann eine sogenannte dynamische Lösung entgegen gestellt werden. Hierbei können die Roboter, die Relaisfunktionen erfüllen, weiterhin bewegt werden. Das heißt, die Relaispositionen ändern sich über die Zeit, oder verschwinden ganz. Die meisten reaktiven Lösungen, die es für die Aufrechterhaltung der Kommunikation in der Literatur gibt, sind solche dynamischen Systeme (z.B. in Mosteo et al. [2008, 2009] und Wagner and Arkin [2004]) . Dabei werden meist Gruppen von Robotern hinter einem Leader-Roboter hinterher geführt. Sie sorgen durch den Aufbau von Ketten dafür, dass der Leader-Roboter weiterhin Kontakt zur Gruppe oder zu einem Leitstand hat. Es sind jedoch auch dynamische Lösungen für nicht-reaktive, also für Planungsverfahren denkbar.

Beide Verfahren haben ihre Vorteile. Der Vorteil der statischen Lösung liegt vor allem in der realen Umsetzbarkeit der Pläne. Die Bewegungskontrolle ist einfacher, da sich Roboter, die einmal am Ziel sind, nicht mehr bewegen. Es müssen keine aufwendigen Methoden implementiert werden, die ein synchrones Bewegen ermöglichen. Zudem ist

ein statischer globaler Mehrroboterplan für einen Menschen einfacher verständlich.
Dennoch haben dynamische Lösungen ebenfalls Vorteile. Dabei ist vor allem zu beachten, dass dynamische Lösungen höchstens so viele Roboter benötigen, wie der statische Fall. Dass sie allenfalls gleich viele Roboter benötigen, ist offensichtlich, da der statische Fall als ein Sonderfall der dynamischen Lösung angesehen werden kann. In der dynamischen Lösungen kann ein Relaisroboter bewegt werden, muss es aber nicht. Allerdings können Umgebungen generiert werden, in denen die dynamischen Lösungen deutlich weniger Roboter benötigen, als die statische Lösung.

Lemma A.1 *Eine statische Lösung der koordinierten Navigation unter spatialen Nebenbedingungen kann beliebig viele Roboter mehr benötigen als die dynamische Lösung.*

Beweis. Sei die Nebenbedingung die Sichtbarkeits-Nebenbedingung, wobei die Sichtbarkeit in der Distanz nicht eingeschränkt ist. Die Abbildung A.1 zeigt eine Umgebung in der die dynamische Lösung besser ist, als die statische Lösung:

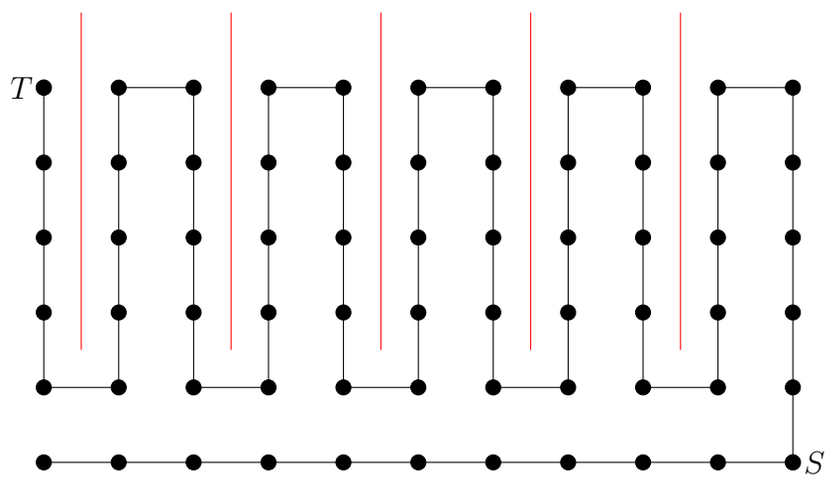


Abbildung A.1: „Kamm-Umgebung“ zum Vergleich dynamischer und statischer Lösungen.

Die eingezeichneten schwarzen Linien entsprechen dem Bewegungsgraph. Der Roboter kann also nur auf ihnen bewegt werden. Die roten Linien sind Hindernisse, durch die undurchsichtig sind. Punkt S ist der Startpunkt, Punkt T der Zielpunkt. Um einen Roboter, ohne die Nebenbedingung zu verletzten, von S nach T zu bewegen, wird bei einer dynamischen Lösung genau ein weiterer Relaisroboter benötigt. Dazu wird der Relaisroboter auf dem unteren Weg synchron zu dem Roboter auf dem oberen Zick-Zack Weg bewegt. So erreicht der Roboter T, ohne die Nebenbedingung zu verletzen (siehe Abbildung A.2).
Im statischen Fall jedoch wird für jeden Bereich zwischen den Hindernissen („Taschen“) ein Relaisroboter benötigt, wie in Abbildung A.3 zu sehen ist.
Damit benötigt die dynamische Lösung konstant viele Relaisroboter, unabhängig von der

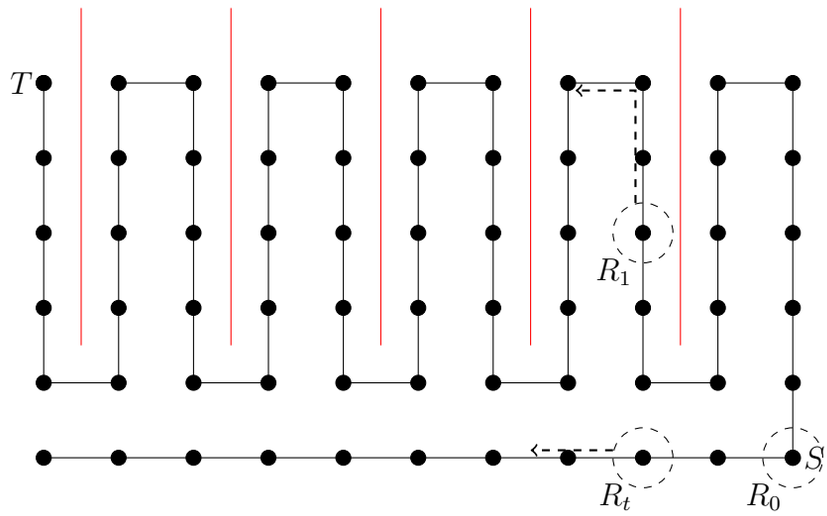


Abbildung A.2: Bei der dynamischen Lösung wird der Relaisroboter R_t synchron zum Roboter R_1 bewegt bis R_1 das Ziel erreicht. Da R_t immer sowohl R_1 wie auch S sieht, wird die Nebenbedingung nicht verletzt.

Anzahl der Taschen, während der statische Ansatz genauso viele Relaisroboter wie Taschen benötigt. Durch Generieren von beliebig vielen Taschen kann die statische Lösung dann beliebig viel schlechter werden als die dynamische Lösung.

■

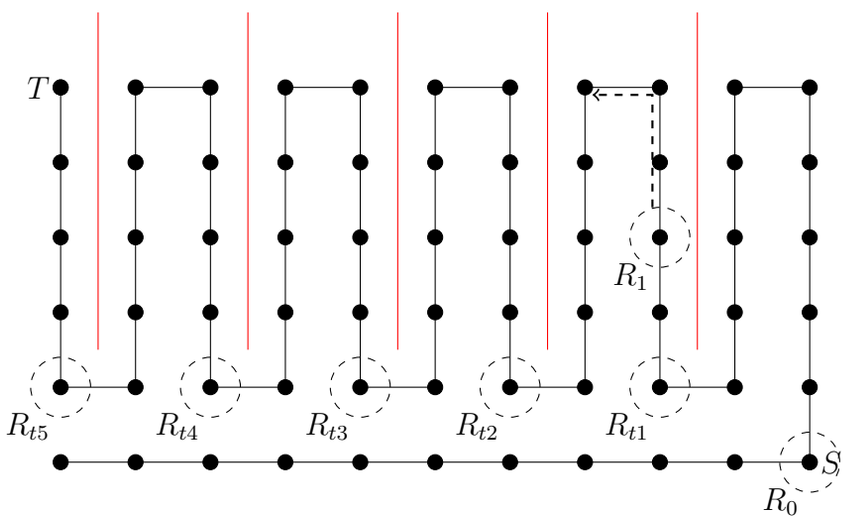


Abbildung A.3: In der statischen Lösung muss jeder Bereich zwischen den Hindernissen beobachtet werden. Daher wird für jeden dieser Bereiche ein Relaisroboter benötigt.

B
Eigenschaften und Beweise

B.1 Eigenschaften des SCG

In diesem Teil des Anhangs werden einige allgemeine Eigenschaften der SCGs gezeigt. Die SCGs sind als schnelle Möglichkeit zum Vergleich von Bewegungs- und Bedingungsgraph geschaffen worden. Das bedeutet, sie sollen primär dazu dienen, die Entscheidung zu treffen, ob eine Bewegung erlaubt ist. Daraus ergibt sich folgende Eigenschaft:

Lemma B.1*: Steht ein Roboter auf v_i, so kann ein weiterer Roboter von v_i zu jedem Knoten, der in SCG(v_i) mit v_i verbunden ist, auf direktem Weg gelangen, ohne die Nebenbedingung zu verletzten.*

Beweis. Annahme: v_i und v_j sind in SCG(v_i) verbunden. Es steht ein Roboter auf v_i. Ein weiterer Roboter fährt von v_i nach v_j. Es existiert kein Weg von v_i nach v_j, auf dem der Roboter nicht die Nebenbedingung verletzt.
Man betrachtet nun den Subgraphen $G^{sub}_{Bew}(v_i)$wie in der Definition beschrieben. Da v_i und v_j in SCG(v_i) verbunden sind, gibt es mindestens einen Weg in $G^{sub}_{Bew}(v_i)$zwischen beiden Knoten. Auf jedem dieser Knoten ist der zweite Roboter aber mit dem ersten Roboter gültig im Sinne der Nebenbedingung verbunden. Dies gilt, da ansonsten dieser Knoten nicht in $G^{sub}_{Bew}(v_i)$enthalten wäre. Damit ergibt sich ein Widerspruch zur Annahme. ■

Dies bedeutet, dass SCGs in dem gewünschten Sinne genutzt werden können, nämlich als schnelle Möglichkeit, alle möglichen gültigen Bewegungen aufzuzeigen. SCGs sind

tatsächlich sogar vollständig, was die Suche nach direkten Wegen von einem Knoten zu einem anderen betrifft.

Lemma B.2*: Es gibt keinen direkten Weg von v_i nach v_j bei dem mindestens ein Knoten des Weges nicht in SCG(v_i) liegt und die Nebenbedingung erfüllt.*

Beweis. Angenommen es gibt solch einen Weg und der Knoten v_x liegt auf dem direkten Weg und nicht in SCG(v_i). Damit der Roboter beim Betreten von v_x die Nebenbedingung nicht verletzt, muss er im Bereich eines anderen Roboters sein. Dieser andere Roboter kann nicht der Roboter auf v_i sein, da sonst v_x in SCG(v_i) liegen würde. Damit benötigt man einen Relais-Roboter, damit die Nebenbedingung beim Betreten von v_x nicht verletzt wird. Daher kann es kein direkter Weg sein, was ein Widerspruch zur Annahme ist. ■

Lemma B.3*: Es gibt keinen direkten Weg von v_i nach v_j, bei dem alle Knoten in SCG(v_i) liegen aber v_i und v_j in SCG(v_i) nicht durch eine Kante verbunden sind.*

Beweis. Die Bedingung für eine Kante in SCG(v_i) zwischen v_i und v_j ist, dass es einen Weg in $G^{sub}_{Bew}(v_i)$zwischen v_i und v_j gibt. Da keine Kante existiert, existiert ein solcher Weg nicht. Es müsste also mindestens ein Knoten außerhalb von SCG(v_i) besucht werden, um v_j zu erreichen. ■

Lemma B.4*: Existiert ein direkter Weg zwischen v_i und v_j, wenn ein Roboter auf v_i steht, ist dies äquivalent zu v_j ist in SCG(v_i) mit v_i verbunden.*

Beweis. v_j ist in SCG(v_i) mit v_i verbunden $\Rightarrow$ Existenz eines direkten Weges. Folgt aus Beweis B.2. Existenz eines direkten Weges von v_i nach $v_j \Rightarrow v_j$ ist in SCG(v_i) mit v_i verbunden: Folgt aus dem Beweis B.3. ■

Lemma B.4 bedeutet für die Anwendung, dass der SCG(v_i) als Look-Up Tabelle für alle existierenden direkten Wege von v_i verwendet werden kann. So sind alle Knoten, die in SCG(v_i) mit v_i verbunden sind, auf direktem Wege erreichbar. Zusätzlich gibt es keinen weiteren Knoten, der von v_i direkt erreichbar ist.
Auch wenn die Nebenbedingungen, die hier vorgestellt werden, symmetrisch sind, kann durch die Gestaltung der Umgebung und damit des Bewegungsgraphen eine Symmetrie bei zwei SCGs nicht angenommen werden.

Lemma B.5*: Existiert ein direkter Weg von v_i nach v_j, bedeutet das nicht, dass es einen direkten Weg von v_j nach v_i gibt.*

Beweis. Vergleiche Abbildung B.1. SCG(v_j) enthält v_j und in SCG(v_j) ist v_j mit v_i verbunden. Daher existiert ein direkter Weg von v_j nach v_i. Umgekehrt ist zwar v_j in SCG(v_i) enthalten aber nicht mit v_i verbunden. Daher gibt es keinen direkten Weg von v_i nach v_j. ■

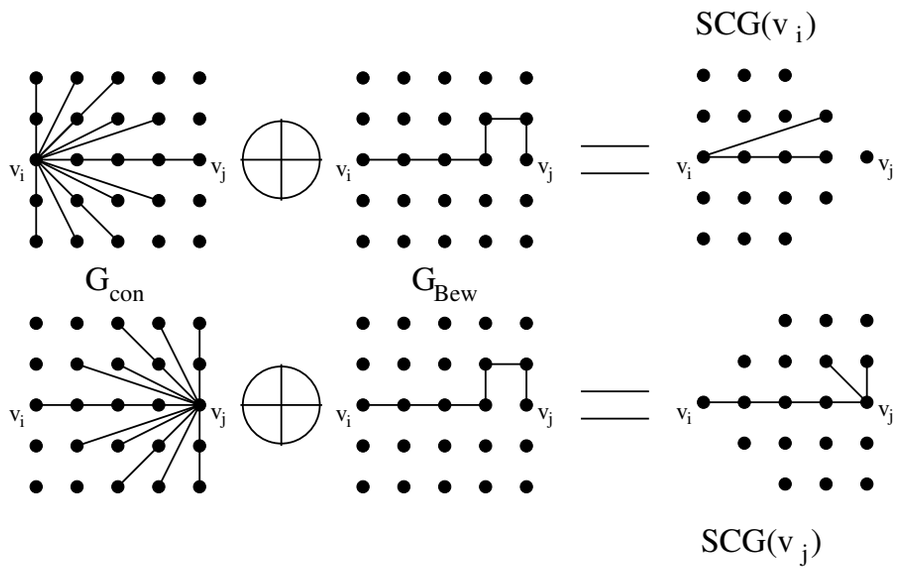


Abbildung B.1: Beispiel für die Asymmetrie zweier SCGs. Oben SCG(v_i), unten SCG(v_j). Eine Kante von v_j nach v_i in SCG(v_j) impliziert dabei nicht eine Kante von v_i nach v_j in SCG(v_i).

Diese „Asymetrie“ ergibt sich dabei aus der Tatsache, dass jedes SCG von einem bestimmten Knoten abhängt. Daher beziehen sich SCG(v_i) und SCG(v_j) mit $v_i \neq v_j$ jeweils auf unterschiedliche Teilgraphen des Bewegungsgraphen. Daher bedeutet eine Kante von v_i nach v_j in SCG(v_i) nicht die Existenz eine Kante von v_j nach v_i in SCG(v_j).

B.2 Korrektheit des Pfadsuchalgorithmus

Theorem B.6*: Ein Pfad von v_a nach v_b der mit Hilfe des Algorithmus 2 gewonnen wurde, verletzt nicht die Nebenbedingung.*

Um dieses Theorem zu beweisen, wird die Ausgabe des Algorithmus genauer betrachtet. Dabei wird angenommen, dass der Algorithmus einen Pfad gefunden hat. Zuerst muss die Situation betrachtet werden, dass der gefundene Pfad nur aus den Knoten v_a und v_b besteht. Dies bedeutet, dass v_b direkt in SCG(v_a) mit v_a verbunden ist.

Lemma B.7*: Wenn ein Roboter auf v_a (Start des Algorithmus) steht und ein Roboter auf v_b (Ziel des Algorithmus), so erfüllen sie die Nebenbedingung.*

Beweis. Die Knoten v_a und v_b stammen aus dem Steinerbaum Algorithmus. Daher sind beide Knoten im Bedingungsgraph verbunden. ■

Ist diese einfache Lösung nicht möglich, berechnet Algorithmus 2 eine Kette von SCGs, auf denen sich die Roboter bewegen. Dabei wird jeweils beim Betreten des Basisknotens (also jenem Knoten der im Pfad des Algorithmus auftaucht) ein temporärer Relais-Roboter positioniert. Diese Kette muss die Nebenbedingung erfüllen, d.h.:

Lemma B.8*: Aufgrund der Konstruktion des Algorithmus bildet die Ausgabe von Algorithmus 2 eine Kette von SCG(v_a), SCG(v_1), SCG(v_2), ..., SCG(v_b), die zusammenhängend ist.*

Beweis. Nach Algorithmus 2 ist die Bedingung für das Anhängen eines Knotens v_y als Blatt zu einem vorhergehenden Knoten v_x, dass v_y noch nicht im Suchbaum enthalten ist und dass v_y in SCG(v_x) mit v_x verbunden ist. Der gefundene Pfad besteht also aus Knoten die bei dem jeweiligen Vorgänger im SCG mit diesem verbunden sind. Daher ist die resultierende Kette aus SCGs ebenfalls verbunden. ■

Um zu zeigen, dass auch der komplette Weg die Nebenbedingung nicht verletzt, muss die Bewegung charakterisiert werden. Das heißt jeder Bewegung wird eindeutig ein SCG zugewiesen. Das dies stets funktioniert sagt:

Lemma B.9*: Jede Bewegung des Roboters von einem Knoten zum anderen geschieht innerhalb eines SCGs.*

Beweis. Sei B eine Bewegung v_i nach v_j, die aus dem Plan nach Algorithmus 2 entstanden ist. Angenommen v_i liegt in SCG(v_1), aber nicht in SCG(v_2), und v_j liegt in SCG(v_2), aber nicht in SCG(v_1). Damit läge die Bewegung nicht innerhalb eines SCGs. SCG(v_1) und SCG(v_2) sind nach Lemma B.8 zusammenhängend. Das bedeutet, v_2 ist in SCG(v_1) mit v_1 verbunden. Demnach kann die Bewegung nicht die Bewegung von SCG(v_1) nach v_2 sein, da v_2 in beiden SCGs vorhanden ist. Der Weg nach v_2 ist allerdings auf der Basis des SCG(v_1) berechnet, d.h. es existieren in ihm nur Wege, die auch vollständig in SCG(v_1) liegen (siehe Lemma B.2). Daher existiert keine Bewegung, die sich nicht eindeutig einem SCG zuordnet lässt. ■

Die vorangegangen Lemma ermöglichen den Beweis von Theorem B.6:

Beweis. Angenommen, der Algorithmus 2 gibt einen Pfad Π zurück, der an mindestens einer Stelle die Nebenbedingung verletzt.
Bedingung für die Ausführung des Pfades ist, wie oben dargestellt, dass auf jedem Knoten, der im Pfad Π existiert, ein temporärer Relais-Roboter positioniert wird. Wenn der Pfad, den die Roboter befahren sollen, die Nebenbedingung verletzt, dann gibt es eine Bewegung von Knoten v_i nach v_j die diese Verletzung auslöst. Jede Bewegung, die der Roboter ausführt, kann einem SCG zugeordnet werden (siehe Lemma B.9). Damit die Bewegung v_i nach v_j im Plan erscheint, muss sie so einem SCG zugeordnet werden können. v_i und v_j liegen in einem SCG auf dem direkten Pfad zum Basisknoten des nächsten SCG. Dieser direkte Pfad existiert (vergl. Lemma B.1) und verletzt nicht

die Nebenbedingung. Daher kann auch der Schritt v_i nach v_j die Nebenbedingung nicht verletzten. ■

Bis hierhin wurde angenommen, dass der Algorithmus 2 einen Pfad von einem Knoten aus der Endpositionierung zum nachfolgenden Knoten findet. Wird so ein Pfad nicht gefunden bedeutet dies, dass es für diese Endpositionierung keine gültige Lösung gibt.

Theorem B.10*: Der Algorithmus SCGPath ist komplett, d.h., wenn es eine Lösung gibt, so findet er auch eine. Existiert keine Lösung so wird auch dies angezeigt.*

Um die Pfade zwischen den Knoten des Steinerbaums zu berechnen, kommt der SCG-Pfad Algorithmus (Algorithmus 2) zum Einsatz. Hierbei ist zu beachten, dass zuerst nur eine Aussage über die Ergebnisse des Algorithmus SCG-Pfad getroffen wird. Nach Theorem B.10 bedeutet das, dass der Algorithmus, wenn es einen Weg zwischen Knoten **a** und Knoten **b** gibt, der die Nebenbedingung nicht verletzt, dann wird er auch gefunden. Ansonsten wird gemeldet, dass keine Lösung vorhanden ist.

Beweis. 1) Angenommen es existiert kein Pfad PI, SCG-Pfad gibt aber einen Pfad aus. Nach Theorem B.6 ist ein Pfad von SCG-Pfad immer ein Pfad, der die Nebenbedingung nicht verletzt. Dies ist ein Widerspruch zur Annahme.
2) Angenommen es existiert ein Pfad τ, SCG-Pfad findet jedoch keinen Pfad.
Betrachten wird in diesem Fall zuerst der Zustand der Suchstruktur $searchtree$. Sie enthält aufgrund Zeile 18 jeden Knoten nur einmal. Zusätzlich bedeutet die Tatsache, dass SCG-Pfad keinen Weg gefunden hat, dass die While Schleife aus Zeile 5 dadurch abgebrochen wurde, dass jedes Blatt in $searchtree$ besucht wurde und v_b als Endknoten von τ nicht in $searchtree$ enthalten ist. Wenn, wie angenommen, τ ein korrekter Pfad ist, so ist er eine Verkettung von SCGs. Daher gibt es einen Knoten v_n, sodass gilt, v_b ist mit v_n in SCG(v_n) verbunden. Damit kann v_n ebenfalls nicht in $searchtree$ enthalten sein, da sonst v_b zu $searchtree$ hinzugefügt worden wäre. Diese Argumentation gilt weiterhin für v_{n-1} als Vorgänger für v_n. Auch v_{n-1} ist nicht in $searchtree$ enthalten. Entlang der SCG-Kette des Pfades τ bedeutet dies letztendlich, dass auch der Ausgang des Pfades, also **a** nicht in $searchtree$ enthalten ist. Dies ist aber ein Widerspruch zu Initialisierung in Zeile 2. ■

Theorem B.10 sagt allerdings nichts über die generelle Lösbarkeit des Problems aus. Das bedeutet, auch wenn der Algorithmus für ein Knotenpaar keine Lösung findet, heißt das nicht, dass keine Lösung für das Gesamtproblem existiert. Lediglich dieser Knoten, zu dem kein Weg existiert, kann nicht in der Endpositionierung liegen. Sollte dieser Knoten jedoch ein Zielpunkt sein, ergibt sich daraus die Unlösbarkeit des Problems.

Abbildungsverzeichnis